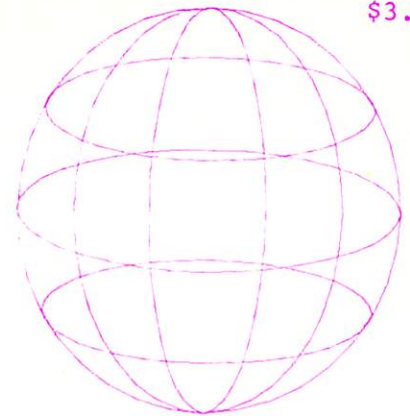


# COMAL 7 TODAY

\$3.95



SPHERE PLOT



COMAL FOR NON-GURUS

FAST LOAD COMAL 0.14

BOOK REVIEWS

DEFINE FUNCTION KEYS

2.0 EXTERNAL PROCEDURES

BATCH FILES

INTERSTELLAR DUST CLOUDS

COMAL DISASSEMBLER

DISK EDITOR

1520 PLOTTER DRIVER

FAST DIRECTORY READER

MAILING LIST MAKER

GEMINI 10X GRAPHICS DUMP

Publisher: COMAL Users Group, U.S.A., Limited  
6041 Monona Drive, Madison, WI 53716  
Editor: Len Lindsay  
Captain COMAL art by: Frank Hejndorf

### **CONTENTS (PARTIAL LISTING)**

How To Type In COMAL Programs - 2  
Illiterates Unite - 6  
COMAL Standards Conference & Graphics Kernal - 8  
COMAL In The Real World, Tom Kuiper - 12  
COMAL Event Of The Year - 15  
COMAL Communique, Ernie McDonald - 16  
Sharing - 18  
Amazing Delete Key - 19  
Redefine Function Keys - 21  
A Simple Exercise For Non Gurus, Colin Thompson - 22  
Fast Loading COMAL - 25  
COMAL 2.0 External Procedures, Captain COMAL - 27  
Batch Files, David Stidolph / Jesse Knight - 30, 32, 58  
Book Reviews - 34  
Interstellar Dust Clouds, Tom Kuiper - 48  
The 5 Byte Integer Problem - 51  
Freeway, W. P. Miller - 52  
Disk Editor, Ian MacPhedran - 56  
2.0 Fast Directory Read, George Jones - 58  
Disassembler in COMAL 2.0, Ian MacPhedran - 59  
Make DATA Statements From Files, T Creasy - 60  
Make COMAL Object File, Ian MacPhedran - 61  
1520 Plotter Driver Routines, Kevin Quiggle - 62  
Gemini 10X Graphics Dump - 66  
Find Radicals, Steve Smullen - 68  
Mailing List Maker, Doug Drake - 69  
Pressure Drops And Water Flow, Lowell Zabel - 71  
Differential Equations, Lowell Zabel - 73  
Disk Directories - 79  
Subscriber Specials - 81

### **ADVERTISERS INDEX**

MARCA Commodore Computer User Fair - 15  
PlayNET - 17  
Transactor - 20  
The Guide - 25  
Sparcug - 29  
Peripherals Plus - 75  
United States Commodore Users Group - 75  
COMAL Users Group, USA, Limited - 76

---

# From the Editor's Disk

---

Are you ready for the **COMAL EVENT OF THE YEAR?** See page 15 for details. If you have a modem, you now can get instant **COMAL support** thru PlayNET every Thursday night. Would you like a week of intensive **COMAL training?** Come to the Lincoln College Commodore Computer Camp, June 23 -28, 1985. Every afternoon I'll be having a long class in using COMAL. There still is room, so call them at 217-732-3155 to register now (and remember to bring alot of blank disks).

We're moving. All the equipment here was getting a bit cramped. Plus next month we hope to get our HP LASER printer. That should allow us to put out a really nice newsletter. Plus we just hired a freelance writer to help. A disadvantage of moving is that now we pay a high monthly rent for the 3 room office.

A common question is "**What book should I get?**". That is hard to answer, especially since there now are 13 different COMAL books. We decided to print reviews of every COMAL book currently available, even if it took a dozen pages to do it. Now armed with this information, you should be able to decide which books are right for you.

**COMAL is ideal for schools.** Five European countries have already adopted it as the programming language to be taught in their schools. It's only natural. COMAL was designed to replace BASIC and to be very useful as an educational language. If you know a school system that has Commodore 64 computers and isn't using COMAL yet, we would like to introduce them to COMAL. Send us the name of the proper person at the school to contact, and we will give them a free subscription to COMAL TODAY. Why should the Europeans remain years ahead of us?

COMAL needs your support as we change the history of how people use their home computers. Share your programs. Send a disk of your working programs. Share your information. Send us an article or note about your latest COMAL discoveries. Try to send it as a text file on disk (we have PaperClip, WordPro, and EasyScript). In return, **we'll send you one of our disks FREE** (just tell us which one). We now are preparing for another SPECIAL TOPIC for COMAL TODAY - peripherals! We need articles and programs about disk drives, printers, plotters, joysticks, light pens, track balls, and more. If you want to be part of the special issue we need to hear from you right away.

IBM is now selling **IBM PC COMAL** (written by the same people who did the C64 COMAL 2.0 Cartridge) in Denmark. The first few months it is only available with a Danish manual. The English manual is expected by the end of summer. Price is about \$250-\$300. Since we are importing it, contact us if you are interested.

We are the official distributor for the C64 COMAL 2.0 Cartridge made by Commodore in Denmark. Move up to the Cartridge now with our special: \$25 worth of COMAL disks and books of your choice FREE with each cartridge. Also, we are offering a quantity discount price to **dealers, user groups, and schools.**

How do you order COMAL? A convenient way is the 800 order line: 800-356-5324 ext 1307. It also is the slowest. They collect orders and mail them to us once a week. You also can order on-line via PlayNET, but the same delays can be expected. The **fastest is to call us** at our office: 608-222-4432. Denise can take your order process it within 2-3 days usually. Or just mail in your order. This is the only way to order if you aren't using a charge card.

Line numbers are irrelevant to a running COMAL program. COMAL only provides line numbers for our benefit in editing the program. Thus most magazines do not use line numbers when listing a COMAL program. It is up to YOU to provide the line numbers. But of course, **COMAL can do it for you** quite easily. Just follow these steps to type in a COMAL program listing:

- 1) Enter command: **NEW**
- 2) Enter command: **AUTO**
- 3) Type in the program
- 4) When done:  
Version 0.14: Hit <RETURN> key twice  
Version 2.0 : Hit <STOP> key

Remember - use unshifted letters throughout entering the program. If letters are capitalized in the listing it does not mean use the SHIFT with those letters. They are capitalized merely for a pleasant, easy to read, look. The only place to use SHIFTED letters is inside quotes. Also, you don't have to type leading spaces in a line. They are listed only to emphasize structures. You **DO** have to type a space between COMAL words in the program.

**LONG PROGRAM LINES:** We are continuing to print COMAL TODAY with two columns per page, printed in elite (12 pitch) with 40 characters maximum per line. This makes it easiest to read. However, some program listings have program lines that extend beyond the 40 character limit. We decided to list these lines in the same manner that COMAL uses when listing long lines on a 40 column screen. We simply break the line right at the 40 character spot, and continue it on the next line, indenting it properly to keep the program structures obvious. These are called wrap lines. To draw your attention to these continued lines we add a //wrap line comment to the end of the line. Whenever you see this make sure you type both lines as one continuous program line! The following example includes a line with more than 40 characters that we must list on two lines, but you must type in as one long program line:

```
IF CURRENT'NAME$<>"STOP" THEN PRINT'LABE  
L(CURRENT'NAME$,PHONE$) //wrap line
```

If you type in this long program line as two shorter program lines, COMAL will not object (although sometimes it will)! But, the program will not work unless it is entered as one long line. The procedure name PRINT'LABEL is split onto two lines in the listing, but the //wrap line draws your attention to this fact.

[illegible]

COMAL TODAY welcomes contributions of articles, manuscripts and programs which would be of interest to readers. All manuscripts and articles sent to COMAL TODAY will be treated as unconditionally assigned for publication and copyright purposes to COMAL Users Group, U.S.A., Limited and is subject to the Editor's unrestricted right to edit and to comment editorially. Programs developed and submitted by authors remain their property, with the exception that COMAL Users Group, U.S.A., Limited reserves the right to reprint the materials, based on that published in COMAL TODAY, in future publications. There will be no remuneration for any contributed manuscripts, articles or programs. These terms may be varied only upon the prior written agreement of the Editor and COMAL Users Group, U.S.A., Limited. Interested authors should contact the Editor for further information. All articles and programs should be sent to COMAL Users Group, U.S.A., Limited, 5501 Groveland Terrace, Madison, WI 53716-3251. Authors of articles, manuscripts and programs warrant that all materials submitted are original materials with full ownership rights resident in said authors. No portion of this magazine may be reproduced in any form without written permission from the publisher. Local Users Groups may reprint material from this issue if credit is given to COMAL TODAY and the author. Entire contents copyright (c) 1985 COMAL Users Group, U.S.A., Limited. The opinions expressed in contributed articles are not necessarily those of COMAL Users Group, U.S.A., Limited. Although accuracy is a major objective, COMAL Users Group, U.S.A., Limited cannot assume liability for article/program errors.

Please note these trademarks: Commodore 64, CBM of Commodore Electronics Ltd; PET, Easy Script of Commodore Business Machines, Inc; Calvin the COMAL Turtle, CAPTAIN COMAL, COMAL TODAY of COMAL Users Group, U.S.A., Limited; Buscard, PaperClip of Batteries Included; CP/M of Digital Research; Z-80 of Zilog; IBM of International Business Machines; Apple of Apple Computer Inc; PlayNET of PlayNET Inc; COMPUTE!, COMPUTE!'s GAZETTE of COMPUTE! Publications, Inc; IBM of International Business Machines. Sorry if we missed any others.

---

# Letters

---

Dear Editor:

After using the 2.0 Cartridge since mid-December, I can find absolutely no weaknesses in the language. It is truly a wonder to work with. I demonstrated it for my local Commodore dealer and he asked me to write a brief on the language which I have enclosed. You may use any or all of it if you wish, even though most parts have already been stated many times by you and others.

I have made some discoveries and encountered some quirks that you may wish to pass on to the readers:

\* You should remind people about the TRACE command. I had forgotten all about the command until I was thumbing through COMAL TODAY #4 and came across it in your article. It has already helped me to reduce the already small debugging time required by COMAL.

\* The PROTECT64 program works very well, however it is also protected (cute!) so that one cannot list it to see how it works. Could you or someone write an article on its mechanics? Is it possible to protect only certain portions of a program?

COMAL is everything you have claimed it is. You have done a great job of documenting the language. Keep up the good work.

K.S. Manitoba

[Ed note: The PROTECT64 program can only protect an entire program. No one has yet broken it. We don't know how it works. As for TRACE, we plan on some notes in next issue.]

Dear Editor:

I would like to point out what still appears to be a need for some people. For science students at the college level who want to start with COMAL or change from BASIC to COMAL, the available books and disks, COMAL TODAY, etc, are not meeting the real need. However, it is true that more experienced programers just take off with it and have the highest regard for COMAL as an excellent computer language. Yet, for example, Kelly's book [FOUNDATIONS] is too wordy. An example of what seems more useful is Jim Butterfield's article on COMAL in the November 1984 issue of COMPUTE. He should write a short introductory book to go with your COMAL HANDBOOK. Jim has a nice concise style of writing. Whatever was referred to in LETTER FROM A CARTRIDGE USER in COMAL TODAY #5 on page 15 (ie., what his wife wrote in the DIFFERENTIAL program on DEMO DISK #2) would be interesting to see in COMAL TODAY. Tom Kuiper's contributions in #5 are also very much appreciated. I also feel good about Colin Thompson's enthusiasm and insight. The need for concise programs on computational topics like standard deviations for sets of random numbers, use of FFT to see how random such generators really are, least square fit, radioactive decay problems, statistics, and dynamics problems is very great. This would help a large number of computer users to break away from BASIC more easily. I trust my suggestions may be constructive and indicators of what could lead to a still greater success for COMAL.

D.O., Professor, University of Minnesota



Dear Editor:

I teach English at a community college and "Introduction to Computers" (using COMAL, of course) at the local adult education shop. I would like to develop a network of COMAL Using Educators so that we could share ideas, projects, procedures, functions, problems, etc. Although I myself am particularly interested in college and lifelong learning, educators at all grade levels are welcome.

One possibility might be to devise a grant proposal involving the use of inexpensive computers (guess which one I mean) with COMAL for disadvantaged students.

If you are at all interested in this and would not mind having your name on a circulating membership list, please write to: Jim Ventola, 328 Poplar Street, Roslindale, MA 02131.

Dear Editor:

I would like to pass along a suggestion. I am a high school teacher and teach several computer programming classes. I would like to teach COMAL. However, our curriculum supervisor will not allow it since he has contacted several area universities who have advised us to stay with BASIC or PASCAL. They have never heard of COMAL and therefore they cannot recommend it for schools. I have written to several area colleges and universities sending copies of your literature believing that if they became aware of what COMAL is, they will encourage its use in secondary and elementary schools.

If you really want to publicize COMAL, here is my suggestion: Send detailed information about COMAL to the computer science departments of colleges and universities. COMAL appears to be little known among educators and yet I see it as an excellent solution to the problems caused by the lack of structure in BASIC and the difficulty of PASCAL.

I hope you find my suggestion helpful. If our area colleges knew about COMAL, it would be much easier to convince our administrators to allow its use.  
R.F. Ohio

Dear Editor:

No COMAL Users Group is listed for Colorado in COMAL TODAY #4, and to my knowledge none exists. This letter explains the reasons for this (as I see them). I'm hoping things will turn around.

I'm a professional engineer and an old Fortran programmer (since 1968). The Commodore C64 seemed to me to be the best home computer around, so when the price finally dropped I went out and bought a C64 system. Once I started working with it, however, I became enraged with the BASIC on it, which seemed to have been added as an afterthought! Once I found out about COMAL I rushed out and sent away for the disk. For the first time, I could run the C64 with an integrated operating system. Compared to BASIC (Boo, Yuk, Blah) COMAL is far more than just a language. BASIC, however, has turned out to be a disoperating system for the C64; some evidence is offered here.

When I finally made it to a Colorado Commodore Computer Club meeting with my COMAL books and disk under my arm, I was in for a nasty surprise. The club had the disk already, but no one was using it. What was going on instead was a process of "software gathering"; people were just collecting and learning to use the software available for the C64, and no one was writing programs! It would seem that writing programs in BASIC is such an ugly process that people have developed an aversion to programming altogether. COMAL makes programming possible on the C64, but the damage has been done. I'm going to make some effort to correct this.

B.O. Colorado

Dear COMAL Users Group:

My grandchildren returned home last Saturday... I must add how fascinated I am becoming with COMAL. It is far superior to my Simon's Basic Cartridge, and I will be quite content to do all my programming in it if it will work with the paintings that I do on HI-RES screens accompanied with interrupt driven machine language music. You see, I am an artist and about a year ago I got a computer with the idea of transferring my art to screen and disk, using slight animation to suggest gentle motion. I am trying to incorporate all this with the ML interrupt driven music from Master Composer by Access.

I had some trouble with the BEGINNING COMAL disk. I believe the problem comes from being "taught" in school to do what it says in the book, and I did. The new directions on the printed sheet said not to LOAD the programs in, but to ENTER. Of course after days of thinking about it, ENTER appends. I would ENTER as told, make my additions to the program as told, then when I would LIST it I would get not only the current program but the program from the original COMAL Intro and MENU. Ach! What a bummer! After doing it umpteen times I learned not to. Wish I were a youngster who forgets what the school says and just "tries different things." I finally do, but only after many defeats.

This was what was happening to me in the CAPTAIN COMAL GETS ORGANIZED. I would "Q" quit the menu program, attempt to type in what the book said and never get rid of the menu. Too dumb to try NEW until I finally found it in a list of COMAL commands. Once I finally got my wits about me, with your most gracious help, things went fairly much OK.

I gave an unplanned for demo at a User meeting the other night and have several folks interested in forming a SIG [Special Interest Group] of COMAL. One of our members teaches Special Ed kids in

our school system and was most impressed with the COMAL turtle. She will probably be getting in touch with you, since I gave her a copy of the C64 disk and it has your address on it.

Corordially, H.S. Florida

Computer tools for computer fools... I make weekly payments to the utility companies and monthly payments to everyone else. My tax refund comes and I can't wait to buy more COMAL. Best wishes, J.B. Michigan.



COMAL TODAY is now distributed internationally and receives submissions from people of all ages. The picture above was submitted by Rhianon Lindsay, age 7. You can see Captain COMAL hard at work by his computer with his favorite COMAL book in his hand.

# Illiterates Unite!

"What's a turtle?"

"It's a sprite."

## "What's a sprite?"

"It's a little creature that runs around the screen."

## "What's a screen?"

Yes, there are a few people left in the U.S.A. (or one for sure) who know nothing about computers. Before becoming a staff member of COMAL TODAY, I thought RUN was a magazine for joggers and BYTE a journal for orthodontists.

Fortunately, the optimistic editor of COMAL TODAY sees this ignorance as a plus. There may be a few readers who are new hands in the field, and documenting my journey through the maze of computer-ese may answer the questions of others.

So, what exactly is a turtle? It's a drawing tool for the computer. The turtle is used the same way a pencil is used on paper, the major difference being no one ever called a pencil a snake or a worm. A pencil is a pencil --- we don't come up with different cute names for it in hopes of purposely confusing people.

The turtle shows up on the screen as a small triangle or arrow. It has a pen attached to it and as you move the turtle around, the pen writes. The turtle is quite docile and responds well to the following commands:

TURTLESIZE: the turtle can be made bigger or smaller in an instant, at the whim of the person at the keyboard. (I think this command should be re-named "turtle-on-a-diet.")

HIDETURTLE: if you're mad at your turtle and don't want to look at him/her, you can command HIDETURTLE and the turtle will disappear from the screen. However, the turtle will continue to draw for you while in this state of exile. He really is one heck of a nice guy.

SHOWTURTLE: When you are ready to readmit the turtle to your kingdom, punch in SHOWTURTLE, and a turtle will reappear on your screen. Don't assume this is the same turtle you had before: he may have tired of your little games and sent in a replacement.

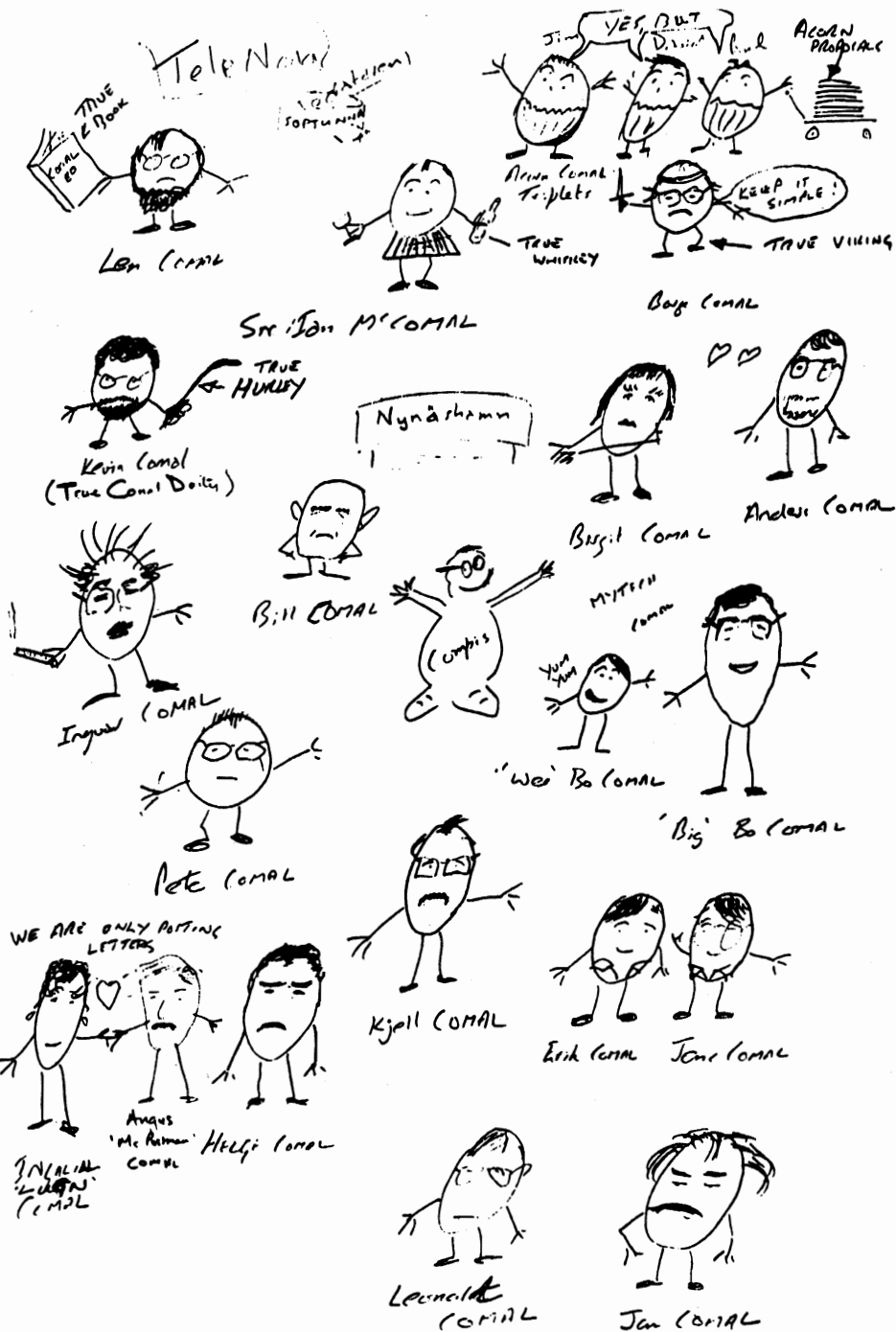
PENUP and PENDOWN: PENDOWN is easy to understand. It simply means the pen is touching the paper, so to speak, and you can draw. But if a turtle's only job is to draw, why do we need a PENUP order? Len came to the rescue and explained it this way: Suppose you drew a house and then wanted to add a sun to the sky. Would you want a line to be drawn between the house and the sun? No. So you need to pick up the pen as you move the turtle. This made good sense, even to me.

Fellow novices, take heart. I learned all of this merely by watching Len and his turtle at play. I didn't touch the computer even once. Actually, I'm not anxious to start fooling around with these suspicious-looking machines. Does that glazed look in the eyes of the computer person appear instantaneously the first time one touches the keyboard? But I will set these worries aside, as duty calls and the adventure begins. (Next month: The Art and Science of Loading a Disk).

[illegible]

A bit of comic relief was provided to all attending the COMAL Standards Meeting in Sweden by Angus, a representative from Scotland. He drew a caricature of each person attending the meeting. See next page.





# COMAL Standards Conference

COMAL manufacturers and representatives of various user groups held the sixth international standardization and development conference March 14-16, 1985. TeleNova, part of Swedish Telecom and manufacturer of the COMPIS computer, hosted the meeting in Nynashamn, Sweden.

COMAL has been designed to combine the simplicity and ease of use of BASIC with the power and structure of Pascal. The language is officially adopted for use in schools in Sweden, Denmark, Norway and Ireland and approved for use in Scotland. It is in the educational field that COMAL has received most attention so far, but with more implementations becoming available it will undoubtedly be increasingly popular as a powerful and elegant general purpose language.

Borge Christensen, one of the inventors of the language, outlined some new ideas from an educational view point. Len Lindsay, leader of the US COMAL Users Group, presented the rapidly growing activities in the United States.

It is hoped that the very cooperative attitude of the participants at the conference will ensure the continued development of COMAL as a standardized language.

COMAL is now available from the following companies:

Acorn for the BBC Microcomputer  
Commodore for the C64 and 8000 series  
Dansk Data Elektronik for the Supermax and UNIX  
IBM for the IBM PC family  
Mytech for various operating systems (MS-DOS, CP/M86 and UNIX)  
Metanic for CP/M operating systems  
Regnecentralen RC for the Piccoline and Partner  
TeleNova for the COMPIS/SCANDIS computer

Several other software houses present at the conference are planning releases of COMAL for many other computers.

## Graphics Kernal

It is proposed that a GRAPHICS KERNAL be adopted as an extension to COMAL. This is a preliminary report from a working group on the subject.

The GRAPHICS KERNAL should allow for graphics primitives to be part of some type of PACKAGE or built into a specific COMAL implementation. Parentheses may or may not be required around parameters.

The user should be allowed to specify their own co-ordinate system - based on REAL numbers. The way to define this is with a WINDOW command:

WINDOW(<left>,<right>,<bottom>,<top>)

The default window is implementation dependent. \*

VIEWPORT is used to tell how the window is mapped onto the screen (normalized). It is a graphics extension - if lacking the whole screen is the default.

Where possible we should try to conform to the GKS standard.

It should be allowed to have two separate screens - one for text and one for graphics - or one screen to handle both at once.

More ▀

There should be two different keywords to use for clearing each screen.

Clear text screen: PAGE or ????  
Clear graphics screen: CLEAR

CLEAR would clear the current viewport (whole screen by default).

An extension should include a keyword to clear the entire graphics screen regardless of current viewport, but the word to use was not decided upon. Possible words included, CLEARALL and CLEARSCREEN.

An extension should allow for splitting a screen into two parts - part text and part graphics. The keyword to use to go into this mode should be SPLITSCREEN. It's syntax and meaning has not been defined. To return to a full graphics screen the word FULLSCREEN should be used.

It is also proposed that two forms of drawing on a graphic screen be allowed:

- 1-Relative and Absolute X/Y coordinates
- 2-Turtle (LOGO compatible when possible)

A current position is maintained in both systems. In turtle graphics it is the turtle's location.

Here are the proposed keywords for X/Y:

MOVE(<x>,<y>)  
MOVE(5,9)

Move, relative to the current position, to a new current position. The example would change the current position by adding 5 to the current x position and 9 to the current y position. There is no change to the screen.

MOVETO(<x>,<y>)  
MOVETO(5,9)

Move the current position to the absolute point specified. There is no change to the screen. The example would move the current position to coordinate 5,9.

DRAW(<x>,<y>)  
DRAW(5,9)

Draws a line from its current position to the new position. It is the same as MOVE except that a line is drawn.

DRAWTO(<x>,<y>)  
DRAWTO(5,9)

Draws a line from its current position to the specified position. It is the same as MOVETO except that a line is drawn.

PLOT(<x>,<y>)  
PLOT(5,9)

It was not determined whether or not this should be in the GRAPHICS KERNAL or its extension. It will plot a point at the specified position and change the current position to that point. It is defined as a MOVETO(<x>,<y>) followed by a DRAW(0,0). If implemented, the word PLOT should be used.

It was not defined what should happen if drawing outside the screen is attempted.

#### TURTLE GRAPHICS:

##### PENUP

If the turtle moves with the pen up, no drawing will take place. It will then be similar to the X/Y MOVE command.

##### PENDOWN

If the turtle moves with the pen down, it will draw a line as it goes. This is similar to the X/Y DRAW command.

It was suggested by the working group that PENUP be the default. However, at this time, PENDOWN could be suggested as well, as it might be expected by BEGINNERS who will be the majority using this drawing system.

More ■

FORWARD <length>  
FORWARD 50

Moves the turtle forward the number of units specified. If the pen is down, a line is drawn.

BACK <length>  
BACK 50

Moves the turtle back the number of units specified. If the pen is down, a line is drawn. The current turtle heading is unchanged.

SETXY <x>,<y>  
SETXY 5,9

Moves the turtle to the specified absolute position. A line is drawn if the pen is down.

LEFT <degrees>  
LEFT 90

Turns the turtles current heading to the left the specified number of degrees.

RIGHT <degrees>  
RIGHT 90

Turns the turtles current heading to the right the specified number of degrees.

SETHEADING <degrees>  
SETHEADING 180

The heading system used by the turtle is composed of 360 degrees. Headings are calculated using MOD 360 and negative headings are allowed. It is proposed that heading 0 be pointing up to the top of the screen and that the heading increase in a clockwise manner. In the example the turtle would then be facing down towards the bottom of the screen.

It is proposed that showing a turtle image on the screen be an extension and if used it should be allowed to turn the image on and off. Proposed words to do this are:

SHOWTURTLE  
HIDETURTLE

If a SETXY command places the turtle outside the screen the result was not defined.

If turtle drawing takes it outside the screen two results are possible, and two words define which result will be used:

WRAP

If WRAP is in effect, drawing off one edge of the screen will cause the turtle to reappear on the opposite edge (like the screen was a cylinder).

NOWRAP

If NOWRAP is in effect, drawing off one edge of the screen will cause the turtle to disappear off the edge. It will draw a line to the edge and then disappear. It's position will be maintained as if it were still on the screen.

BOTH X/Y and TURTLE

Both drawing systems will share common commands such as:

PENCOLOR(<color index>)

This sets the current color of the drawing pen.

BACKGROUND(<color index>)

This sets the current background color of the screen.

Also, it is proposed that three functions be included in the GRAPHICS KERNAL that will return information about the current position:

XCOR

Returns the current X coordinate

YCOR

Returns the current Y coordinate  
More

## HEADING

Returns the current heading (turtle graphics only)

=====

At this point time ran out. Other words should be included as part of the GRAPHICS KERNAL and extensions. These could include:

HOME  
FILL  
CIRCLE  
ARC  
BORDER  
PLOTTEXT ■

## DELAYS, DELAYS, DELAYS

Overall, most COMAL items have come out as scheduled, or nearly so. But there have been problems -- and some problems still remain, such as:

The book STARTING WITH COMAL has still not been released by its publisher, Prentice Hall. It is now a year late.

The 8096 COMAL 2.0 system has still not been released. It is finished. The reason for not releasing it is that the Danish company wishes to copy protect it and have not figured a good way to do so yet.

BEGINNING COMAL has been a favorite book, despite problems. We now are out of stock again, and getting more seems to literally take MONTHS. We have 600 copies on order. If the British publisher ships them to us we will be in good shape for awhile.

If you have ordered items (such as the above) and find the delay too long, please send us a note asking to cancel the order. We will be happy to send you a refund. Meanwhile, lots of COMAL products are in the works, but to avoid the DELAY problem, we aren't announcing them early. Hope you understand.

## COMAL CARTRIDGE IS SMART

We already told you how COMAL lists your programs neatly, indenting the blocks and displaying the keywords in UPPER case and variable names in lower case. But, did you think what that would look like if the C64 were in UPPER/GRAPHICS mode? All the lower case would be shown as upper case, and the upper case as graphic symbols, right? Wrong. COMAL is too smart for that. While your computer is in UPPER case mode, COMAL lists BOTH keywords and variables in upper case! Try it. Load in a fairly long COMAL program. Now do this:

> LIST

> Press the space bar to pause the listing after a few lines. Then press SHIFT and the Commodore Key. Notice the lower case switch into upper case and the upper case into graphics.

> Press the space bar again and the listing will continue. Now notice that both keywords and variables are upper case.

> Press the space bar to pause the listing once more. Press SHIFT and the Commodore Key again. Notice the cases switch again.

Now, the reason I am making such a fuss about this small but nice feature is that COMAL lists the program like this while listing to disk as well as to the screen. That is important to know! If you plan to transfer a COMAL 2.0 program to be used with COMAL 0.14, you must LIST it to disk and then ENTER it into COMAL 0.14. But, remember that COMAL 0.14 doesn't like upper and lower case listings. So before you list the program, just press SHIFT Commodore key to switch to the all upper case listing mode. Problem solved. Thanks to Joel from LoadStar for the tip.



# COMAL in the Real World

by Tom Kuiper 1985

[ED NOTE: The letter attached to this article included this: "I was away for somewhat over a month, carrying out observations from Kitt Peak (Arizona) and from the NASA airborne observatory. The latter involved a lot of programming and gave me the idea for this article. I hope some of your readers will react to it, and that you can arouse some interest in it in Denmark:"]

Borge Christensen describes on page 8 of COMAL TODAY 1 how, around 1972, he became disillusioned with BASIC as a language for teaching computer programming. Niklaus Wirth's book SYSTEMATIC PROGRAMMING, which introduced PASCAL to the world, inspired him and his colleagues to develop a language which blended the convenient interaction with BASIC and the structures of PASCAL. Care was taken to preserve what Christensen describes as the "human interface". Today, besides communicating with humans, computers are increasingly used for sensing and control in the home: thermostats, heaters, air conditioners, water sprinklers, security sensors, telephone answering, video equipment, etc. Programs used for this are designed to process data as soon as the measurements are complete (in "real time"). This presents the challenge to develop another interface -- the "real world" interface -- to the same level of excellence. By this I mean a COMAL optimized for interacting with sensing and control devices, laboratory and home electronics. The characteristics of such an interface can be derived from experience in the laboratory and factory. Since 1973, I have used a variety of languages to control instruments in observatories. Just as BASIC and PASCAL were developed for teaching, and each embodies some of the features desirable

in that application, so also a number of languages have been optimized for some aspects of data acquisition and instrument control.

At first, assembly language was used for software modules which were intended to interact with specialized instruments: signal generators, position and angle encoders, switches, special displays, pressure and temperature sensors, spectrum analyzers, etc. The advantage of assembly language is that is close to the language of the computer itself -- the machine code-- and so can be very efficient and fast. It can easily be tailored to any task of which the computer is physically capable. Such modules are normally linked into a larger program also compiled from assembler or perhaps FORTRAN. The drawback to this approach is that any change, however minor, requires a new compilation and linkage, which is a time consuming process. Programmers working in this area became adept at "patching", which is changing the machine code in the computer directly to avoid having to rebuild the program from scratch.

By 1973, Chuck Moore had developed FORTH to control the equipment of the National Radio Astronomy Observatory 36-ft telescope at Kitt Peak, Arizona. Simple tasks can be defined in assembly language at the keyboard, compiled on the spot, and linked to the body of the program without relinking the whole program. Such a module is labelled with a single descriptive word, and can then be executed simply by typing the word at the keyboard. More importantly, complex tasks can be defined in terms of sequences of these words. Thus a programmer can develop a "word" to control some device, say a fast analog-to-digital converter, and test it by itself. If it doesn't work  
More ▀

right, he can instruct FORTH to FORGET it, and try a new version. Once he gets it right, he can combine it with other such "words" to perform a useful task, testing it in turn until it is right. All of this is done interactively, which is an enormous improvement over working with programs which must be entirely compiled and linked each time. FORTH spread like wildfire from observatory to observatory, and soon to other places where custom software was needed. One drawback is that FORTH is completely different in structure and philosophy from most other languages. Also, it is easy, and quite common, to write FORTH source code which is totally unreadable.

A different approach was taken in the middle '70s by Hewlett-Packard in developing its proprietary HPL, a BASIC-like language designed for instrument controllers. Being a manufacturer of instruments as well as computers, HP ensured that commands existed in HPL to handle data transfer to and from instruments efficiently. In addition, they added a very powerful feature: the ability to halt a program, or to let it come to a halt through an error, to change it, and then to let it continue from where it stopped! When you are flying an experiment at 45000 ft, minutes away from a scheduled turn, have just spent fifteen minutes getting data from a newly developed instrument, and the program halts because of a divide by zero, you don't have the option of repeating your measurement. A quick fix is just what is needed. A drawback of HPL, however, is that data manipulation is done at the speed of the interpreter which, though fast compared to others, is still very slow compared to compiled languages. Also, HPL shares many of the weaknesses of traditional BASIC so that HP Technical Basic, which has much in common with COMAL, has replaced it in HP's latest generation of desktop computers. HP Technical Basic, however, is slower than HPL.

What are the characteristics of these languages which facilitate custom software developed for instrument control and data gathering? Over the years, these are what I have found valuable or wished for:

1) INTERPRETED CODE - Having an interpreter or, better yet, a line-by-line compiler, saves enormous amounts of time in testing hardware interfaces and data manipulation software. This, of course, is a feature of COMAL.

2) ERROR RECOVERY - This is the ability to change the offending line when an error is detected, and then to continue execution from the corrected line. This also saves a lot of time when testing. It is a rare feature among interpreters, because it requires rigid separation between code and data, including all internal pointers. HPL also allows a kind of global TRAP ... HANDLER ... ENDTRAP capability. Near the beginning of the program, one specifies:

ON ERROR GOTO error-module

The error module can then take action appropriate to ERN, the error number, and continue from the line where the error occurred:

CONT ERL

or any other line, as appropriate. This capability does not exist in COMAL and may be the hardest to implement [see note 1].

3) INTERRUPT HANDLING - A device will set a bit ("raise a flag") when it requires servicing. If it is a high speed device, it may need immediate response. A structure of the form:

ON INTERRUPT FROM device EXEC service is needed, where the service routine is specified as a PROCedure. This is the key feature of "real time" computing [see note 2].

4) DMA I/O - Direct memory access input/output frees the program from having to process every word or byte exchanged with a device. Instead, a  
More

5) LIVE KEYBOARD - A running program does not normally respond to a keyboard input unless it requests one. In some cases, it may interrupt the program and return temporarily to the operating system. HPL allows you to type in and execute instructions while the program is running. It does this by inserting the typed-in line between the current and next lines of code being executed. This allows you to examine and change variables, for example, without halting the program. You can use the computer as a calculator, without disturbing the running program. You can also respond to a number input request from the program with a formula. For example, if an input in degrees Celsius is required, but your thermometer reads 69 deg F, you can enter:

6) COMPILABILITY - With an interpreter, the relatively slow speed becomes apparent when a sequence of operations is repeated many times. A great increase in speed can be obtained by turning this part of the program into a machine-language subroutine. At the very least, one should be able to replace it with a module compiled from assembler. This feature exists in COMAL 2.0 Packages. In an ideal world, one would be able to compile Packages from a variety of common languages, such as FORTRAN, C, and PASCAL.

8) MACHINE INDEPENDENCE - With the great variety of computers now available, one should not have to rewrite programs to change computers. The same software should be executable on computers ranging from simple controllers to mainframes. COMAL is getting there (see COMAL IMPLEMENTATIONS by Kevin Ryan in COMAL TODAY #5, p.6).

ED NOTE: Wonderful article! We are forwarding a copy to the COMAL Standards Group.

ED NOTE 2: COMAL 2.0 for the CBM 8096 by Unicomal includes INTERRUPT which performs this function for interrupts on the IEEE-488 bus. See COMAL HANDBOOK second edition page 168 for an explanation of this implementation.

[illegible]

User Groups may include any of our disks in their disk libraries for use by their members. The exception is the matching disks to COMAL books. These are copyright by the publishers and may be provided only to users who also own the companion book.

## THE COMAL EVENT OF THE YEAR

Mark your calendar now. If you are interested in COMAL, you will definately be interested in this. The last weekend in July, COMALites from around the world finally will meet just 18 miles from downtown Philadelphia. It's the MARCA Commodore Computer Users Fair -- and you are invited. Here is just some of the reasons you won't want to miss this:

\* Meet the father of COMAL, Borge Christensen. The man who started it all will be making his first appearance in the United States at this show. This is the man behind two favorite COMAL books: BEGINNING COMAL and COMAL FROM A TO Z. Straight from Denmark, he's bringing a COMAL controlled robot. You won't want to miss that!

\* Come to the special COMAL sessions. Three different presentations by Borge Christensen, Len Lindsay, and Mindy Skelton. Find out what's going on. There is limited seating so register early for these sessions.

\* See the new COMAL products. We have some spectacular things planned to introduce at the show. This will be the first time you can see them. Plus, you can look over all the other COMAL disks and books at our large double size booth right by the main entrance.

\* Talk to the authors you've been reading, including: Len Lindsay (COMAL HANDBOOK, COMMODORE 64 GRAPHICS WITH COMAL, CAPTAIN COMAL GETS ORGANIZED), Borge Christensen (BEGINNING COMAL, COMAL FROM A TO Z), Jesse Knight (COMAL 2.0 PACKAGES), and Mindy Skelton (CAPTAIN COMAL'S GRAPHIC PRIMER).

\* Of course, COMAL is only part of the show. Save some of your time to visit the Commodore booth to see their new products. And most software companies will probably be demonstrating their new programs as they prepare for the Christmas market.

This will be the largest Commodore Show ever. It's up to all of us to make sure that COMAL is well represented. We'll be there. We hope you will too.

You're invited to the  
biggest party at  
Valley Forge since  
George brought  
the boys!



# M.A.R.C.A.

The biggest Commodore User Fair in the US.

## July 26, 27, 28

Valley Forge Convention Center, Valley Forge PA

- Speakers! • Seminars! • Hanging out!
  - Fun! • Vendors! • Great Buys!
  - Social Events! • Fun!
- Areas Tours available.

Meet the names you've only read about. Jim Butterfield. Dick Immers. Len Lindsey. Many, many more! Ask the questions you need answers to. Have 2½ days of non-stop Commodore fun! Bring the whole family. Lots to do. See. And buy. Bargains galore!

Pre-registration by July 1: 2½ days \$25 M.A.R.C.A. Members \$15 Family Rates available.

For pre-registration information: M.A.R.C.A., P.O. Box 1902, Martinsburg, West VA 25401.

## DON'T MISS THE PARTY!

# COMAL Communique

By Ernie McDonald / April COD

Load up COMAL. Remember:

```
LOAD "boot*",8 <RETURN>
```

Here we go on Lesson Two:

While it's loading, let's chat a moment -- but don't forget to answer the prompts from your language disk!

One of the great strengths of COMAL is its modular construction...those little blocks called procedures. They're easily spotted in any COMAL program listing because the language does a great job of indenting them for easy identification. Well, this -- and a couple of other things -- is the subject at hand.

In keeping with our policy of writing only useful programs, this one is meant to solve many of Life's little -- or major -- problems. And be able to blame somebody else!

I'll explain things as we go along. So type "new" <RETURN>, then "auto" <RETURN>, and we'll get started. As usual we'll start out explaining what we're going to do.

```
0010 //Pasadena Commodore Computer Club
0020 //COMAL Lesson two
0030 // Subject: Procs
0040 // Decisions, Decisions
0050 // * Start of Program * //
0060 dim whether$ of 78
0070 question
0080 decision
0090 result
0100 //
-----
```

// line 60: the "dim" allocates space for strings. In our example you can enter a response up to 78 characters.

Lines 70-80: Think of a menu. These are the procedures we're going to use.

```
-----
0110 proc question
0120 print chr$(147)//clear screen
0130 print "What yes or no decision"
0140 print "do you want me to make",
0150 input whether$
0160 print
0170 print "How many of my bits do"
0180 print "you want me to poll",
0190 input numpoll
0200 print
0210 favor:=0
0220 against:=0
0230 endproc question
-----
```

// Line 110: Procedure header, called from "menu." Must be concluded with endproc command. Line 140: note comma. If left off, question mark falls to next line on screen. Line 210: zeroes string. Note ":= ". In COMAL it means "value assigned to." A "=" means equal to. Test-run program so far, if you like.

```
-----
0240 //
0250 proc decision
0260 for poll:=1 to numpoll do
0270 bits:=rnd(0,1)
0280 if bits=false then
0290 against:=against + 1
0300 else
0310 favor:=favor + 1
0320 endif
0330 endfor poll
0340 endproc decision
0350 //
-----
```

// Line 260: simple loop. If you forget the command "do" COMAL adds it. Line 270 "RND" command creates random numbers from 0 to 1. Line 280: "false" is a command. It equals zero. ("True" always equals 1).

More ▀



Line 300: "Else" command speaks for itself. Line 330: End of loop. Type in "next" and COMAL corrects you!

```
-----
0360 proc result
0370  if favor>against then
0380    print "Yes! Sure! Absolutely!"
0390  elif favor<against then
0400    print "I'm sorry! The majority of"
0410    print "my bits say no! But like"
0420    print "all good scientific polls,"
0430    print "I can be manipulated"
0440  else
0450    print "You're on your own, buddy!"
0460    print "We have a bunch of"
0470    print "fencesitters inside here!"
0480  endif
0490  print
0500  print "The results were:"
0510  print
0520  print "For the decision: ";favor
0530  print "Against the idea: ";against
0540 endproc result
-----
```

In this Proc we test the conditions with "if." If not true, we drop through to the "elif" (means "else if"), and if that's not true, we test with "else." "If" statements can be made up many ways: if,then; if,then,endif; if, then, else, endif; if, then, elif, endif; and if, then, elif, else, endif.

-----  
There's the program. Run it by typing "run" <RETURN>. It will convert you from being a procrastinator to a programmer! ■

## FILES USED

C64 COMAL allows you to use file numbers 1 through 255. However, some file numbers are used by the system. You should not use these file numbers to avoid conflict:

COMAL 0.14 uses file number 255 for PRINTER access and file number 1 for system disk use.

COMAL 2.0 uses file number 255 for printer access, file 254 for system use, and file number 253 for SELECT INPUT.

## RANDOM FILES BUG REVISITED

by Jim White

I'm a COMAL 0.14 user who normally uses random files. I have experienced the "write" problem with them too (see COMAL TODAY #6, page 38). I have found that opening and closing the file each time a record is written helps, but the best aid is forcing the file to be written away from tracks 17 and 19. If I have problems, I rename the old file and recreate it so it is forced to another track. The problem seems to occur only when the file is located adjacent to the directory/ block allocation map (track 18). A REQUEST: If possible, I hope someone can furnish a routine to emulate the SETRECORDDELAY command of COMAL 2.0, which automatically inserts a delay when writing to a Random file.

## PlayNET FREE!

The fastest growing full-color Commodore 64 telecommunications network is making the COMAL User Group U.S.A. a VERY SPECIAL OFFER!! Just renew your COMAL Newsletter (10 issues) or Today's Disk subscription and you can join PlayNET FREE (saving subscription fee of \$39.95).

Talk to the "COMALITES" who are already on PlayNET to see what you've been missing;

- Thousands of people to meet,
- News of the latest innovations for C64,
- Weekly on-line COMAL users forum hosted by Captain COMAL,
- Public Domain software,
- National COMAL Bulletin Board,
- and Much, Much More!!

For more information read the article about PlayNET in COMAL Today, Issue # 6, Page 46... and...contact...

COMAL USER GROUP U.S.A.  
TODAY!

(Offer Expires 7/31/85)

# Sharing

by Unknown Author / April COD

The following scenerio probably takes place hundreds of times each day. Perhaps you even have played a part in it!

The scene is the computer section of some discount department store, a large or small computer store, or one of the "software only" organizatons that are becoming more and more prevalent. The time is lunch time but it could be almost any time. There are several customers milling around the software displays "shopping". Several others have disregarded the "do not touch" signs - have stopped the auto run demos and are trying to get the display computers to do various things, but are not succeeding.

Behind the counter is our hero, the salesperson. Now not many people know this, but Steve, our salesperson, arrived to fill this lofty position of computer expert about five weeks ago. He was in the automotive department before - and prior to that it was housewares.

Our computer expert, after five weeks, is not only expected to know EVERYTHING you always wanted to know about your computer, but EVERY peripheral, program, and problem for every computer system the store handles. Now, as if this were not enough, Steve also handles the sales, stocks the shelves and sweeps the floor for "his" department.

Surprisingly, Steve is not doing a bad job. In fact, he's doing a great job considering how understanding and patient all of the customers are!

I can see you all nodding your heads and laughing, because you have been to this store, or one just like it - right? I know I have. I know Steve.

While in his store last week, exchanging my now famous B/I 80 column card - for the third time, I played a part in this scenario. Steve was being asked a lot of questions about programs: was this check program better than this other one? Why won't this program load? Why can't I use the 1541 demo disk to check the drive? etc, etc. I could see that he wasn't certain about some of the answers.

Without realizing it, I was fielding questions and talking to some of the customers, and soon I had a small crowd around me. I talked to about 10 people that afternoon, helped some and got a new member for our group. (I always ask if they belong to a user group!)

Now, not all computer stores, or computer store employees are like this, but these scenes are played very frequently. I remember how appreciatative I was for help when I got stuck - I still am.

The whole point of this story is sharing our knowledge and helping others is a lot of fun. Try sharing -- you'll like it!

## FREE COMAL TODAY TO USER GROUPS

We try to keep informed about what is going on all around the country. What better way than to read the newsletters from User Groups. So let's exchange newsletters. Have your User Group put us on its mailing list and we will do the same for them. It doesn't matter how big your group is. If your group is already exchanging subscriptions with us, watch the expiration date. Renew us for another year and send us a postcard advising us to renew your group as well.

## AMAZING DELETE KEY IN COMAL 2.0

David Tamkin not only discovered that an interesting thing about the delete key - but also a use for it. The delete key will delete the character just to the left of the cursor. We knew that. But what happens when it hits the left edge of the screen at the start of the line? Then it deletes the character under the cursor instead. Of course the entire line still shifts over once to the left. Try it now.

Now, here is a use for the amazing delete key. First, let's assume we are just writing a quick little program and we want to save it frequently as we write it. We could put these two lines at the start of the program:

```
0010 // DELETE "0:TEST"
0020 // SAVE  "0:TEST"
```

Now if you list those two lines, you can erase the line number and remark to first delete the previous TEST file, and then save the new version. This saves time, since you don't have to type the DELETE and SAVE commands each time. Now, here is an even easier way. Try it - you might not believe it the first time. Enter the following two lines exactly as shown. Where you see [RVS] hold the CONTROL key down and press the 9 key (that is reverse field on). Then the 7 t's you type next will all be in reverse field. After those t's you see [OFF], which means hold the CONTROL key down and press the 0 key (that is reverse field off). Make sure you type all the quotes exactly:

USE SYSTEM

```
DEFKEY(1,"EDIT"13""13""145"[RVS]ttttttt
[OFF]"13"[RVS]ttttttt[OFF]"13")
```

Now clear the screen and try pressing function key 1. I hope you had a disk in your drive. If you did, the program is now saved on it. Magic? No. Just a precise set of instructions all in one function key. Here is what it does:

1) Issue the EDIT command ("EDIT"13")

The EDIT command will list the first line of the program and place the cursor just after the line number, ready for you to edit the line. The 13 represents the carriage return which is CHR\$(13). Remember, CHR\$(13) is equal to ""13"" in COMAL 2.0.

2) Issues another carriage return ("13")

This accepts the first line shown by the EDIT command. COMAL then lists the next line, placing the cursor just after its line number.

3) Issues a cursor up ("145")

This moves the cursor back up to the first line - still just after the line number.

4) Issues seven DELETE keys ("[RVS]ttttttt[OFF]")

The delete key is represented as a reverse field t. So those 7 reverse field t's are treated as 7 delete keys. This first deletes the line number on the left. Then pulls in and deletes the // remark from the line.

5) Issues a carriage return ("13")

Since there is no line number on the line, this turns off the EDIT mode. Plus the carriage return executes what is on the line, which is the DELETE command.

6) Issues seven DELETE keys ("[RVS]ttttttt[OFF]")

This time the cursor starts at the beginning of the line. But the same thing is accomplished, deleting the line number and the // remark from the line.

7) Issues a carriage return ("13")

This executes the line, which saves the program.

# MORE THAN JUST THE TIP

Subscribe!  
to:

## The Transactor

The Tech/News Journal For Commodore Computers Vol. 4

...AND GET MORE FROM YOUR  
COMMODORE 64 AND VIC 20

...CHECK THESE  
FEATURES:

- Up-to-date: news, new products and services.
- Insights and straight talk from respected experts like Jim Butterfield.
- Interesting bits of info shared among Commodore users.

SIMPLY THE BEST. Subscribe now and you'll wonder how you managed without it.

THE TRANSACTOR  
SUBSCRIPTIONS DEPT.  
500 Steele Avenue,  
Milton, Ontario, L7T 3P7  
Telephone: 876-4741

Year's subscription  
(6 issues) \$15.00  
(Separate issues \$2.95 each)  
Back issues available.

## REDEFINE FUNCTION KEYS

IBM COMAL 2.0 and C64 COMAL 2.0 Cart

You can define and redefine what the function keys do from within a running COMAL program. Each function key can be defined to be any string of text. For example:

```
DEFKEY(11,"NO ANSWER") // C64
DEFKEY(41,"NO ANSWER","N/A") // IBM
```

This sets up function key 1 to be the same as typing "NO ANSWER" (plus the IBM version would use N/A as the display on the status line). This can be useful as a reply to an INPUT request:

```
INPUT "What do you think? ": reply$
```

Hit F1 and NO ANSWER appears as your reply. Hit RETURN and that assigns "NO ANSWER" to the variable REPLY\$.

This feature can be very useful. For example, if you wrote a program to process orders, and had to allow for 2 levels of discounts on each item. The exact discount could vary from item to item for each level, so the program couldn't just take a set percentage off the price at each level. First your program would ask for the item name, then the quantity, then the price. Just before requesting the PRICE, you could set up the first 3 function keys with the prices at each level. Then to reply, the user need only press F1, F2, or F3. Here is how you would do it:

```
IBM:
p:=14.95
DEFKEY(41,STR$(p)+CHR$(13),"Full")
DEFKEY(42,STR$(p*.9)+CHR$(13),"10%")
DEFKEY(43,STR$(p*.8)+CHR$(13),"20%")
INPUT "Price: ": price
```

```
C64:
p:=14.95
DEFKEY(11,STR$(p)+CHR$(13))
DEFKEY(12,STR$(p*.9)+CHR$(13))
DEFKEY(13,STR$(p*.8)+CHR$(13))
INPUT "Price: ": price
```

The IBM version would utilize the status line to show what level each function key was at. Both set up the F1, F2, and F3 keys to the price at each level. Now to give the customer 10% discount (level 2) merely press the F2 key in reply to the INPUT request - no need to hit RETURN after it. Of course this is a simplified example, but it should illustrate how the function keys can make programs easier.

We now utilize this feature in our own order processing program and find it makes order entry much faster with fewer errors.

The following program demonstrates using the function keys for price entry to allow discount prices. Run the program. First give it a the List Retail Price (100 for example). Then it will ask you for the selling price. If you wish to sell it at 25% discount, just press the F5 key and COMAL does the rest for you. Try it:

```
// delete "demo.fkeys"
// list "demo.fkeys"
USE system // program for C64 COMAL 2.0
REPEAT
  INPUT "List Retail Price: ": list'price
  price'fkeys(list'price)
  PRINT "Use function keys for entry:"
  PRINT "F1 = full price"
  PRINT "F3 = 10% discount"
  PRINT "F5 = 25% discount"
  PRINT "F7 = 50% discount"
  PRINT
  INPUT "Selling price: ": price
  PRINT "price is";price
  PRINT
UNTIL list'price=0
//
PROC price'fkeys(p)
  clear'fkeys
  defkey(11,STR$(p)+CHR$(13)) //full
  defkey(13,STR$(p*.9)+CHR$(13))//10% off
  defkey(15,STR$(p*.75)+CHR$(13))//25%off
  defkey(17,STR$(p*.5)+CHR$(13))//halfoff
ENDPROC price'fkeys
//
PROC clear'fkeys
  FOR temp:=11 TO 18 DO defkey(temp,"")
ENDPROC clear'fkeys
```



# A Simple Exercise for Non-Gurus

by Colin Thompson

Is this you?

You have 3.4 COMAL books. You have 5.7 COMAL program disks. You have been telling yourself for 4.8 months that you want to sit down and learn about COMAL.

You are "average", whatever that means. You are in the same boat as a lot of other would-be COMALites, clutching your life preserver, looking for that "perfect" time to sit down and try your hand at writing a COMAL program.

```
*****
* COLIN'S MOTTO IS *
*****
* DO IT NOW *
*****
```

So, let's do it now. You don't need to wrestle with a textbook, or tutorial disk. This little project should take you no more than 20 minutes to complete. When done, you will have written a COMAL program.

DO THIS NOW:

1. Read the rest of this article
2. Load COMAL into your computer
3. Pick up the printed article and begin reading from here

FORESIGHT AND PLANNING ARE KEYSTONES TO WRITING A GOOD COMAL PROGRAM.

Let's decide what the program will do, before we go any farther.

Our little program will print some patterns of characters on the screen. Two different patterns may be "painted", depending on a the VALUE of a RANDOM NUMBER.

The COLOR of the characters should vary.

The COLOR of the characters must NOT be the same as the SCREEN color.

The program must LOOP, or REPEAT itself until you press the SPACE BAR.

\*\*\*\*\*

That sounds pretty simple to me, but if you have never written a program, it might be a tall order. Here's how I fill Tall Orders:

```
*****
* BREAK UP THE PROJECT *
* INTO LITTLE PIECES *
*****
```

Roll up your sleeves, We're about to type something. Assuming COMAL is loaded, let's clear the screen by pressing the SHIFT key and the CLR/HOME key at the same time. The CURSOR will be in the upper left hand corner. That's the HOME position of the CURSOR.

Now, look at the letters on the screen. If they are in UPPER CASE (like THIS), you're ready to begin. If not, press the SHIFT key and the COMMODORE key, at the same time. Watch what happens to the letters on the screen. (The COMMODORE key is marked with the Commodore logo - C=. The Pinky finger on your left hand will find it for you.)

All the words and letters you type on the screen should be in UPPER CASE, regardless of how they look on this printed page.

Well now, with all those forbidding warnings out of the way, we're ready to do the fun stuff.

More ➡

Let's think about our proposed program for a minute. We wanted to print one of two different patterns on the screen, in different colors. That means the program must make some decisions. The decisions will be based on the VALUE of a RANDOM number. RANDOM means just that. We can tell COMAL to "make up" a number, within a range of numbers and make it available to us for evaluation. The 16 COLORS the C-64 can make are numbered 0 through 15. 0 is Black, 1 is White, etc.

Let's have COMAL make up a number for us between (and including) 0 through 15.

We will use the COMAL "keyword" called RANDOM. Here's what the "line" would look like:

```
0010 color= rnd(0,15)
```

That is a "line" of a program. The Keyword is RND, meaning RANDOM. "0010" is a line number. All lines must have a different number.

Following the line number we see the word "color", followed by "=". Read it aloud: "Color equals". That sounds like one half of a simple math statement, doesn't it? Well, it is. The other half of the statement is "rnd(0,15)". Let's read it all, now: "Color equals a random number between 0 and 15".

What you just said aloud is how COMAL interprets that line. Now, WHAT is "COLOR"?

Color is a WORD that means a NUMBER. Any COMAL word that represents a number is called a VARIABLE. The VALUE (number) of the word might vary, hence the name VARIABLE. Sometimes COLOR might equal the VALUE of 12 and other times it might equal the VALUE of 7. You see, it varies.

Doesn't that line, when read aloud, sound like a sentence? Sure does. A COMAL program is a series of "sentences" that tell the computer what to do. You write the sentences, and the computer EXECUTES the sentences, starting with the sentence that has the smallest line number.

The programming slang for sentence is "program line" - a line of a program.

As we begin to write our program, we will be typing a series of program lines on the screen. At the end of each line we will press the RETURN key. What happens if you make a typing mistake? You just fix it, using the cursor keys and the INSert/DELeTe key. Then press RETURN.

It's now time to "key in" the little program. First we will erase any COMAL program in memory by typing:

new

Then turn on COMAL's automatic line numbering feature by typing:

auto

(Did you remember to press RETURN?)

Good. Now the screen will say "0010" and the cursor will be one position past the line number. We are ready to start typing in the following program. I suggest you simply start copying the lines and make corrections later. Here's the program. When you've finished typing it in, press RETURN twice to stop the line numbers from coming.

One LAST caution: You don't have to key in the leading spaces. This "indentation" is done automatically for you by COMAL. [the NMN & MNM in lines 140 & 150 should be SHIFTED.]

More

```

0010 background 1
0020 //
0030 repeat
0040   paint'a'pattern
0050 until key$= chr$(32)
0060 //
0070 proc paint'a'pattern
0080   print chr$(19), chr$(142),
0090   color:=rnd(0,15)
0100   if color=1 then color:=7
0110   pencolor color
0120   border color
0130   if color<9 then
0140     for i#:=1 to 331 do print"NMN",
0150   else
0160     for i#:=1 to 331 do print"MNM",
0170   endif
0180   print " comal",
0190   for delay:=1 to 3500 do null
0200 endproc paint'a'pattern

```

How did you do? Did you find the "NMN" and "MNM" did not print as letters? Good. They are supposed to print as graphic characters. These are the slanted bars you see on the right.

We are ready to RUN the program, but first let's check our typing accuracy. Move the cursor to the bottom of the screen and type

list

Your program will be printed on the screen and should match the printed listing you are holding in your hand. If you see an error, move the cursor to the error and type over your mistake. Press return to tell the computer you made a change to the program. Now list the program again. Look OK? Good. Type

run

and watch the program do it's thing. If the screen is filling up with a neat pattern every three seconds, then it's working. The pattern's color should match the border color.

Let's make a change to the program. End the program by pressing the SPACE BAR (that is the CHR\$(32) in line 50). List the program. Change the value of the DELAY in line 190 from 3500 to 300 and then RUN the program again. What happened? Line 190 is simply a time delay. Try other numbers. 1000 equals about one second.

Now change line 10 to say

```
0010 background 7
```

Run the program and wait a while. Did you notice that sometimes the screen remains yellow and no pattern is printed? Well the pattern DID print, but it printed in yellow, so it couldn't be seen. To correct this, change line 100 to say

```
0100 if color=7 then color:=1
```

Notice that we swapped the 1 and 7. Now when line 100 tests to see what color the pattern will print, it will change yellow to white.

Change the " COMAL" to something else. See what happens.

If you have a printer, turn it on. Let's list your program to the printer so you can have a hardcopy. Type this:

```
select "lp:"
list
```

Press return after each line. Try varying the value 331 to something else. What happens? Why?

If you had some trouble typing in the program, and can't make it work, then put the TODAY DISK #7 in the drive and load the program from the disk like this:

```
load "comal'program"
```

(Don't include the ",8" that BASIC requires.) Was this fun? Did you enjoy writing a program? I hope you did. Do you want to try it again? Good. ■

## KWIK LOADING COMAL 0.14

by Randy Finch

In issue #6 of COMAL TODAY someone asked about a boot program for TURBO 64. I do not have this program but I do have KWIK-LOAD! by Datamost. Use the following method to install and use this program on your COMAL 0.14 system disk.

- 1) Load Kwik-Load in the normal manner.
- 2) Load a monitor program that does not reside in the \$CD00-\$CFFF region of memory. Supermon-64 works nicely.
- 3) Save memory area \$CD00-\$D000 on your COMAL 0.14 system disk with a file name of "KL".
- 4) Type in the BASIC program listed below and save it with a file name of "CBOOT".
- 5) To load COMAL 0.14 simply type:  
LOAD "CBOOT",8  
RUN

The boot program will load Kwik-Load and in turn load COMAL 0.14. Overall, the load time is about 25 seconds.

```
BASIC program "CBOOT"
10 IF A=0 THEN A=1 : LOAD "KL",8,1
20 SYS 12*4096+13*256
30 POKE 631,13
40 POKE 632,13
50 POKE 198,2
60 PRINT "[CLR][DOWN][DOWN]LOAD"CHR$(34)
  )"BOOT*",8":REM WRAP LINE
70 PRINT "[DOWN][DOWN]RUN[HOME]";
```

## TURBODISK COMAL LOADER

by Jim White

In COMAL TODAY issue #6 there was a request for a boot program for use with TURBO 64 and COMAL. Such a program follows.

In the boot program there is a reference to TURBODISK.OBJ. This is a program published by COMPUTE! magazine. I use it to load COMAL 0.14 in 15 seconds. The full load sequence of the boot process is

about 30 seconds, including the final chain to my system menu. Replacing the name TURBODISK.OBJ with TURBO 64 and SYS 52224 with the proper address of TURBO 64 should load and initialize it. The percent sign (%) is the load instruction for DOS 5.1E, an enhanced version also published by COMPUTE!, but the version of DOS furnished with the 1541 disk drive should work just as well.

The following BASIC boot program may be loaded by typing:

```
LOAD "*",8,1 (SHIFT + RUN/STOP)
```

More


*The Guide*  
A Monthly Publication For  
Commodore Owners

Formerly "The Northwest Users Guide"

Offering a unique approach to computer  
education and support—with a personable,  
and even humorous touch.

Commodore News and Information  
Programming Tutorials—Beginning and Intermediate  
Software/Hardware Reviews  
COMAL Support

And follow the continuing adventures of COMPU-DUCK  
(found only in THE GUIDE)



Send today for a complimentary copy,  
or send \$15.95 for a One-Year  
subscription to:

*The Guide*  
3808 S.E. Lycintra Court  
Milwaukee, OR 97222  
(503) 654-5603

To use this command, the boot program must be the first program listed in the directory -- otherwise the program name would have to be used in place of the asterisk. Either way, typing ",8,1" after the name followed by the shifted RUN/STOP key will perform the same function as COMAL 0.14 "CHAIN" command (ie., load and auto run).

```
10 REM BOOT DOS AND COMAL
20 PRINT CHR$(147),CHR$(14)
30 LOAD "DOS 5.1E",8,1
40 POKE 53070,8: SYS 52224: POKE 53070,2
50 %TURBODISK.OBJ
60 POKE 53070,8: SYS 49152: POKE 53070,2
70 POKE 198,9 : REM NUM KEYSTROKES
80 POKE 631,ASC("C")
90 POKE 632,ASC("H")
100POKE 633,ASC("A")
110POKE 634,ASC("I")
120POKE 635,ASC("N")
130POKE 636,34 : REM "
140POKE 637,ASC("H")
150POKE 638,ASC("I")
160POKE 639,13 : REM RETURN KEY
170LOAD "C64 COMAL 0.14",8,1
180POKE 53070,8: SYS 2064: POKE 53070,2
```

This program:

- 1) Clears the screen and switches to lower case.
- 2) Boots DOS and inits it.
- 3) Boots TURBODISK.OBJ and inits it.
- 4) Sets up a chain to HI in key buffer. (used after COMAL is loaded)
- 5) Boots COMAL and inits it.

COMAL then executes the CHAIN command that was left in the keyboard buffer. The program HI is loaded and executed. ■

## INDUS GT FASTLOAD

A reader sent us this note: The fastload utility that comes with the Indus GT drive loads COMAL 0.14 and runs a short HI program in 20 seconds.

## PRINT LARGE LETTERS

by Malcolm Bridgham

This routine accesses the character ROMs and prints a selected character to the screen using the bit pattern. Modify it for printer output and you've got a quick banner maker.

```
// delete "0:prnt'lrq'chars"
// save  "0:prnt'lrq'chars"
//
// print large characters
// transactor magazine
// written sep'84 -cz
// converted to COMAL 2.0 by
// Malcolm Bridgham
//
USE system
USE font
textcolors(5,5,1)
DIM char$ OF 8
DIM e(0:8)
FOR i#:=0 TO 7 DO e(i#):=2↑(7-i#)
//
PAGE
LOOP
  INPUT "character set(0-1): ": set
  INPUT "character num(0-255): ": c
  PAGE
  PRINT "CHR$ of";c;"is";CHR$(c)
  textcolors(5,5,1)
  PRINT "Character ROM shape is:"
  PRINT
  print'char'image(2+set,c)
  PRINT
ENDLOOP
//
PROC print'char'image(bank,num)
  getcharacter(bank,num,char$)
  FOR i:=0 TO 7 DO
    line:=ORD(char$(i+1))
    FOR j:=0 TO 7 DO
      IF line BITAND e(j) THEN
        disp:=2
      ELSE
        disp:=1
      ENDIF
      PRINT " *"(disp),
    ENDFOR j
    PRINT
  ENDFOR i
ENDPROC print'char'image
```



# COMAL 2.0 External Procedures

by Captain COMAL

COMAL 2.0 allows procedures and functions to be EXTERNAL. While the COMAL HANDBOOK provides details on this (see pages 342-346 and 128-131), several subscribers had questions on the topic, so we feel general notes would be helpful to all COMAL 2.0 Cartridge users.

NOTE: In this article, we will refer only to PROCEDURES - but the same applies to FUNCTIONS as well.

An external procedure is just a regular closed procedure saved to disk. It can have parameters and even be called from direct mode. There is one restriction: it cannot include any IMPORT statements. First lets demonstrate an EXTERNAL procedure in action - then show a way around the limitation.

Everytime an external procedure is called from a program (or direct mode) it is first retrieved from disk - then executed. Due to this disk access - it may not be wise to make all your small procedures EXTERNAL. However, for our examples in this article - we will use short ones - since they tend to be clearer.

Now - we need a procedure, preferably with a parameter. Let's create a procedure that allows us to pass it a string of characters - and it prints that string twice - with one space in between each printing. We will call it DOUBLE:

```
PROC double(c$) CLOSED
  PRINT c$;c$
ENDPROC
```

That was quite easy. But don't just read this - try it yourself. Do this:

```
Type: NEW    // clear out old program
      AUTO    // start auto line nums
```

Now type in the 3 lines above. Then hit the STOP key. There you have it. Yes, I know - it isn't an external PROC. It is just an ordinary normal PROC. Have patience.

Now, test the procedure. To do this, simply SCAN (or RUN) it and then call it from direct mode - like this:

```
SCAN
DOUBLE("ABC")
```

This is the result printed on your screen:

```
ABC ABC
```

Now, you have a procedure and you have tested it. So, let's make it an external procedure (always test your procedures before you make them external):

First - verify that our procedure fits the requirements of an EXTERNAL procedure:

It is CLOSED. CHECK.  
It contains no IMPORT statement. CHECK.

Now let's make it into an EXTERNAL procedure:

```
SAVE "ext.double"
```

That's it. You can use any filename you wish - though it is HIGHLY recommended that you precede the name with EXT. to make it clear what the file is.

Using an external procedure is equally simple. All that is needed is a procedure HEADER line in the program - COMAL takes care of the rest. Here is an example of using our newly created DOUBLE procedure:  
More

```

text$:="" ; valid$="aeiou"
vlength:=LEN(valid$)
FOR x:=1 TO 18 DO
  text$(x):=valid$(RND(1,vlength))
ENDFOR x
double(text$)
//
PROC double(c$) EXTERNAL "ext.double"

```

Type: NEW // get rid of old program  
 AUTO // provide line numbers  
 type in the program as shown.

Save your program first - then try it out:

```

SAVE "TEST'EXTERNAL"
RUN

```

```
aeaiuiaooooeaiuoiae aeaiuiaooooeaiuoiae
```

Notice that when you ran the program, your disk was accessed. Each time the program wants to execute the DOUBLE procedure, COMAL will retrieve it from disk. This happened only once in our example since DOUBLE is called only one time.

So, you now have used an EXTERNAL procedure. Now, let's play with our program. Let's change the program part into a procedure and call it MAKE'DOUBLE. Let's have the variable VALID\$ be a parameter - allowing us to change the string of characters to be used when we call it. Here is what to do:

Make a new first line:

```
PROC MAKE'DOUBLE(valid$)
```

Also on the first line after it, delete the valid\$ assignment. The line now should just be:

```
text$:=""
```

Then right before the blank remark line add the ENDPROC:

```
ENDPROC MAKE'DOUBLE
```

Here is what the program now looks like:

```

PROC make'double(valid$)
  text$:=""
  vlength:=LEN(valid$)
  FOR x:=1 TO 18 DO
    text$(x):=valid$(RND(1,vlength))
  ENDFOR x
  double(text$)
ENDPROC make'double
//
PROC double(c$) EXTERNAL "ext.double"

```

Now, a RUN of this program would do nothing - it is composed only of procedures. We need to now call it from direct mode or add some program lines. In direct mode lets call it with:

```
MAKE'DOUBLE("aeiou")
```

This should give similar results to the previous program. Now, let's complicate things. Let's make MAKE'DOUBLE a closed procedure. Easy - just add the word CLOSED to the end of the PROC MAKE'DOUBLE header line.

But ... now the procedure doesn't work. Since it is CLOSED we must tell it ALL the procedures we will be calling from the outside program. We can do this with an IMPORT statement as the first line in the procedure:

```
IMPORT DOUBLE
```

Now the program will once again work fine. But now, as it stands, we couldn't make the procedure MAKE'DOUBLE into an external procedure since it contains an IMPORT statement. I promised to tell you a way around this limitation - and now I will:

Use nested procedures.

Remember, to use an external procedure you merely include it's PROC header line. Simply include this PROC header line INSIDE any other EXTERNAL procedure that needs to execute it. So, if we move the PROC DOUBLE header line from OUTSIDE of the MAKE'DOUBLE procedure into the INSIDE

More

of it, once again it will work - and can be made external (also remember to delete the IMPORT statment). Here is how it looks with the change:

```
PROC make'double(valid$) CLOSED
  text$:=""
  vlength:=LEN(valid$)
  FOR x:=1 TO 18 DO
    text$(x):=valid$(RND(1,vlength))
  ENDFOR x
  double(text$)
  //
  PROC double(c$) EXTERNAL "ext.double"
  //
ENDPROC make'double
```

Now, let's make it external. Remember how? Just a normal SAVE:

SAVE "ext.make'double"

We now have TWO different external procedures on the disk:

```
EXT.DOUBLE
EXT.MAKE'DOUBLE
```

Now, let's show how your programs can become quite small in size when you use external procedures. A DEMO PROGRAM:

```
Type: NEW // get rid of old programs
      AUTO // start auto line numbers
      type in the following:
```

```
MD("ABCDEFGF")
//
PROC MD(C$) EXTERNAL "EXT.MAKE'DOUBLE"
```

That's it. Now try: RUN

It worked! Yes - you just witnessed nested EXTERNAL procedures. And in addition - did you notice that in our 3 line program, we called the procedure MD instead of MAKE'DOUBLE? Yes - COMAL will allow you to change its name like that - but the number of parameters must remain the same.

That's it for now. Play around a bit - you may find EXTERNAL PROCEDURES quite exciting. ■

## COMAL WEEK LONG WORKSHOP

This is your chance to experience COMAL. Get a concentrated dose of COMAL and have fun at the same time. The third annual Lincoln College Commodore Computer Camp will be held in Lincoln, Illinois from June 23 to June 28, 1985. I have just checked, and there still is room for you to register. Pick two subjects, one in the morning, one in the afternoon. Then be amazed at what you can accomplish in the evenings. Len Lindsay will be the instructor for the COMAL course held in the afternoons. Of course all 2,000 COMAL programs will be there for use by all 'campers'. If you are interested, call right away -- 217-732-3155.

## Subscribe Today!

To The Southeastern United States  
fastest growing Commodore Users  
Group tabloid newspaper

## SPARKPLUG!

Published monthly by the  
Spartanburg Commodore Users  
Group (SPARCUG)

☐ \$12.00 Per Year U.S.

☐ \$20.00 Per Year U.S.  
SPARCUG Associate  
Membership

☐ \$1.00 For Sample Copy

NAME: \_\_\_\_\_

ADDRESS: \_\_\_\_\_

CITY: \_\_\_\_\_ STATE: \_\_\_\_\_ ZIP: \_\_\_\_\_

Send Check or Money Order To:

**SPARCUG**

P. O. Box 319  
Spartanburg, S.C. 29304.

# Batch File Use

by David Stidolph

One of the new features of the 2.0 cartridge is the ability to use information from a SEquential file on the disk drive as though it was typed in from the keyboard. These disk files are referred to as "batch" files (this is one of the sacred words created in the era of mainframes). COMAL batch files have many uses, but so far the only published examples have been some function key definitions. The following program was written to show a more sophisticated use of batch files.

This program prints names and addresses contained in DATA statements, and allows you to add to the DATA statements easily, without having to add program lines. It can be called SELF-MODIFYING. You'll notice that there are no data statements in this listing. None are necessary. Just type in the program, RUN it and follow the instructions. NOTE: the program writes to the disk drive, and if it cannot, an error will occur, and the program will stop, so make sure you don't have the write protect notch covered.

The program gives you three options: view the names and addresses present in DATA statements, add new names and addresses, and end the program.

The first option allows you to view all the names and addresses and pauses at the end. To slow the listing, press the CTRL key on the left side of the keyboard.

To add a new name and address choose option 2. You will be asked for information on the person (name, address, city, state, and zip). The information is written to a disk file, in the form of DATA statements with line numbers.

When the program ends, it first saves itself, including any updates.

```
// delete "batch'example"
// save  "batch'example"
// by David Stidolph

//a 2.01 COMAL program for the C64
//
DIM name$ OF 20
DIM addr$ OF 25
DIM city$ OF 20
DIM state$ OF 10
//
IF write'protect("0:") THEN
  PRINT "The write protect tab MUST be"
  PRINT "off the disk for this program"
  PRINT "to work. Please remove the"
  PRINT "write protect tab or transfer"
  PRINT "the program to another disk."
  END ""
ENDIF
//
LOOP
  PAGE
  PRINT "This program maintains a list"
  PRINT "of people in DATA statements,"
  PRINT "which the program adds, NOT"
  PRINT "the user."
  PRINT
  PRINT "1) List names"
  PRINT "2) Add name"
  PRINT "3) Quit (and save updates)"
  PRINT
  INPUT "Enter choice: ": c
  CASE c OF
    WHEN 0
      END "" // silent quit
    WHEN 1
      list'names
    WHEN 2
      add'name
    WHEN 3
      quit
    OTHERWISE
      NULL
  ENDCASE
ENDLOOP
More
```

```

PROC list'names
  PAGE
  WHILE NOT EOD DO
    READ name$
    READ addr$
    READ city$
    READ state$
    READ zip
    PRINT name$
    PRINT addr$
    PRINT city$;",";state$;zip
    PRINT
  ENDWHILE
  PRINT "Press any key for menu"
  WHILE KEY$<>CHR$(0) DO NULL
  WHILE KEY$=CHR$(0) DO NULL
ENDPROC list'names
//
PROC add'name
  PAGE
  INPUT "Name      ":" name$
  INPUT "Address   ":" addr$
  INPUT "City      ":" city$
  INPUT "State     ":" state$
  INPUT "Zip Code  ":" zip
  PRINT
  PRINT "Please wait"
  add'name'to'data
ENDPROC add'name
//
PROC add'name'to'data
  // remove old command file
  DELETE "bat.commands"
  // open new command file
  OPEN FILE 1,"bat.commands",WRITE
  // print a zero to have the program
  // quit quietly (during SELECT INPUT)
  PRINT FILE 1: "0"
  // print name and address in form of
  // data statements
  PRINT FILE 1:"9000 data","",name$,""
  PRINT FILE 1:"9010 data","",addr$,""
  PRINT FILE 1:"9020 data","",city$,""
  PRINT FILE 1:"9030 data","",state$,""
  PRINT FILE 1: "9040 data";zip
  PRINT FILE 1: "9050 //"
  // now move new lines to lower line
  // numbers and continue program
  PRINT FILE 1: "RENUM"
  PRINT FILE 1: "RUN"
  CLOSE FILE 1
  // execute commands
  SELECT INPUT "bat.commands"
ENDPROC add'name'to'data

```

```

PROC quit
  DELETE "bat.commands"
  DELETE "batch'example"
  OPEN FILE 1,"bat.commands",WRITE
  PRINT FILE 1: "0" // quit silently
  PRINT FILE 1: "SAVE ""batch'example"
  CLOSE FILE 1
  SELECT INPUT "bat.commands"
ENDPROC quit
//
FUNC write'protect(drive$) CLOSED
  // this function tests whether
  // or not the disk in the drive
  // can be written to. a garbage
  // file name is used.
  //
  filename$:=drive$+"acegikmoq246"
  TRAP
    OPEN FILE 79,filename$,WRITE
    CLOSE FILE 79
    DELETE filename$
    RETURN FALSE
  HANDLER
    CLOSE FILE 79
    RETURN TRUE
  ENDTRAP
ENDFUNC write'protect █

```

## COMAL 2.0 MODEM UPDATE

Donald Pipkin sent in this change to the modem'get\$ function printed in COMAL TODAY #5. It is shorter and ignores errors.

```

FUNC modem'get$(file'number) CLOSED
  DIM c$ OF 1
  old'lo:=PEEK($0324)
  old'hi:=PEEK($0325)
  TRAP ESC-
  TRAP
    POKE $0324,PEEK($032a)
    POKE $0325,PEEK($032b)
    c$:=GET$(file'number,1)
  HANDLER
    c$:=""
  ENDTRAP
  POKE $0324,old'lo
  POKE $0325,old'hi
  TRAP ESC+
  RETURN c$
ENDFUNC modem'get$

```

## BATCH FILES FROM MEMORY

by Jesse Knight

Sometimes batch files come in handy. But they have one drawback. They have to be written to disk or, if you like even slower peripherals, cassette.

There is a way to have COMAL carry out commands, in a similar fashion, using data in memory, instead of from a disk file. The method for doing this ties in to the function key handler. So you can understand this better I will outline how COMAL handles the function keys.

The data for the function key definitions is stored in the RAM at \$DC00-\$DDFF. Up to 32 bytes may be used for each definition. The first 8 definitions are used outside a running program. The last 8 are used inside a program.

When one of the function keys is pressed, the address for the definition is calculated and stored at \$C866-7, which is called KPNT. Next, the length of the definition is copied from the 16 byte table at KEYLEN (\$C855-C864) to KLEN (\$C865).

When COMAL wants to get a character from the keyboard it checks KLEN. If it contains a zero, the first character in the keyboard buffer is used. If KLEN doesn't contain zero, the next character of the definition is used. This continues until all the characters of the definition have been used.

To create and execute a batch file from memory you can use the procedure listed below:

```
PROC perform'string(task$,flag#) CLOSED
FOR x#:=1 TO LEN(task$) DO
  POKE 49151+x#,ORD(task$(x#))
ENDFOR x#
POKE $c866,0 //lo byte
POKE $c867,192 //hi byte
POKE $c865,LEN(task$) //length
IF flag# THEN STOP
ENDPROC perform'string
```

The commands to be executed are passed in TASK\$. Two things to remember are: first, separate the commands with a carriage return, CHR\$(13), and second, a maximum of 255 characters can be handled at one time.

FLAG# indicates if the program should STOP. If it is true, the program will be stopped, and the data in TASK\$ will be treated as commands. If FLAG# is false, the program will not stop, and the data in TASK\$ can be used by INPUT statements that come later in the program.

There are two example programs using this procedure on TODAY DISK #7. PERFORM'DEMO1 uses it to perform commands. PERFORM'DEMO2 uses it to respond to input statements. Both are listed below:

Program Listing - PERFORM'DEMO1:

```
IF TRUE THEN setup //must be first line!
POKE $0804,$43 //enable call to setup
ZONE 5
PRINT
PRINT "x",func'name$
PRINT
FOR x:=1 TO 5 DO
  PRINT x,func'x(x)
ENDFOR x
END "end of program"
//
PROC setup CLOSED
  POKE $0804,$44 //disable call to setup
  IMPORT perform'string
  DIM work$ OF 255
  // next two lines are one long line:
  INPUT AT 24,1,40: "type equation below
    (y=log(x),y=x^2,etc)": statement$
  FOR x#:=1 TO 4 DO
    READ a$
    work$:=a$
  ENDFOR x#
  work$:=statement$+"13"
  FOR x#:=1 TO 4 DO
    READ a$
    work$:=a$
  ENDFOR x#
  work$:=work$+statement$+"13"
```

More



```

FOR x#:=1 TO 3 DO
  READ a$
  work$:+a$
ENDFOR x#
perform'string(work$,TRUE)
//
DATA ""147"del func'x"13""
DATA "del func'name$"13""
DATA "auto"13""
DATA "func func'x (x) closed"13""
DATA "return y"13""
DATA "endfunc func'x"13""
DATA "func func'name$ closed"13""
DATA "return"
DATA "endfunc func'name$"13""
// erase line number
DATA ""20""20""20""20""20""
DATA "run"13"" // now run program
ENDPROC setup
//
PROC perform'string(task$,flag#) CLOSED
  FOR x#:=1 TO LEN(task$) DO
    POKE 49151+x#,ORD(task$(x#))
  ENDFOR x#
  POKE $c866,0 //lo byte
  POKE $c867,192 //hi byte
  POKE $c865,LEN(task$) //length
  IF flag# THEN STOP
ENDPROC perform'string
//
FUNC func'x(x) CLOSED
  y:=SIN(x)
  RETURN y
ENDFUNC func'x
//
FUNC func'name$ CLOSED
  RETURN "y=sin(x)"
ENDFUNC func'name$

```

#### Program Listing - PERFORM'DEMO2:

```

INPUT "demo mode (0=no) ": demo'mode
IF demo'mode THEN setup
//
INPUT "name: ": name$
INPUT "address: ": address$
INPUT "city: ": city$
INPUT "state: ": state$
INPUT "zip: ": zip$
PRINT ""13""
PRINT name$
PRINT address$
PRINT city$," ", " ",state$," ",zip$

```

```

END "end of program"
//
PROC setup CLOSED
  IMPORT perform'string
  DIM work$ OF 255
  FOR x#:=1 TO 5 DO
    READ a$
    work$:+a$
  ENDFOR x#
  perform'string(work$,FALSE)
  //
  DATA "COMAL Users Group"13""
  DATA "6041 Monona Drive"13""
  DATA "Madison"13""
  DATA "WI"13""
  DATA "53716"13""
ENDPROC setup
//
PROC perform'string(task$,flag#) CLOSED
  FOR x#:=1 TO LEN(task$) DO
    POKE 49151+x#,ORD(task$(x#))
  ENDFOR x#
  POKE $c866,0 //lo byte
  POKE $c867,192 //hi byte
  POKE $c865,LEN(task$) //length
  IF flag# THEN STOP
ENDPROC perform'string

```

## BUSCARD & COMAL 2.0

For some unknown reason, if you use both the BUSCARD and the COMAL 2.0 Cartridge, the RUN/RESTORE key combination will crash your computer. If you have a BUSCARD and COMAL Cartridge, try it:

Press the RUN/STOP key and the RESTORE key at the same time. Do this 3 times. Your computer should crash (stop working). To recover, merely turn the computer off then back on.

## BUSCARD COMAL MONITOR

If you have both a BUSCARD and the COMAL CARTRIDGE, you can jump into the Buscard monitor directly from COMAL at any time. Just issue the command: SYS 1006.

COMAL FROM A TO Z  
BORGE CHRISTENSEN  
64 PAGES -- \$6.95

David Stidolph: Ten years ago Borge Christensen defined COMAL. Since then his language has been re-defined and expanded. C64 COMAL 0.14 is a good implementation, and this book is a good manual for it. The book covers the main language as well as the graphics and sprite commands (in this respect "COMAL From A to Z" is ahead of the textbooks on COMAL). Like The COMAL Handbook, keywords are listed alphabetically (sprites and graphics each in their own chapters) with descriptions and examples, although it does not go into as much detail.

There is no index, the table of contents does not list the keywords (for quick look up), and there is no introduction to sprites or graphics. Overall the book is a good reference/tutorial for COMAL 0.14, especially when you consider the price. This would be a fine book for students in the classroom.

COMAL WORKBOOK  
GORDON SHIGLEY  
69 PAGES -- \$6.95

David Stidolph: Using this workbook is one of the best ways to learn COMAL 0.14. It is designed to work with the TUTORIAL DISK, but can be used without it. The book is broken up into sixteen "exercises" that cover most of COMAL, except for the graphics commands. Each exercise has many questions, but no answers are printed anywhere in the book; this is good for schools, but hard on

naming conventions, and some information on sorting, make the book valuable beyond the disk. This book/disk is a useful tool in a programmers set of tools, but it is not particularly valuable to beginners.

Len Lindsay: Once you understand modular programming, this book is a real time saver. It includes over 140 small modules ready to be used in your own programs UNCHANGED! And, to make it even easier to use, all the modules are on the disk that comes with the book - ready to MERGE with your programs. Kevin Quiggle did an admirable job compiling these routines. Other books may be good references and tutorials - this book is USEFUL!

**JESSE KNIGHT**

**108 PAGES -- \$19.95 / \$12.95**

Geoffrey Turney: This book is for all the knowledgeable people out there who know what they are doing at machine code level. It explains COMAL down to its last bit. A tremendous amount of work has gone into discovering the secrets of COMAL 2.0 which are revealed in this book. This includes all the memory addresses for the start of routines such as FLOATING POINT MULTIPLY or HOW TO READ A CHARACTER FROM A FILE. It includes charts and symbol tables as well as a COMAL 2.0 memory map. Plus the disk includes a Machine Language Monitor that works with the cartridge and the C64SYMB file. I hope Jim Butterfield forgives me for calling him (at night) while he was in Trinidad to ask him if his SUPERMON program could be used on the disk once it was adapted for use with COMAL. He said yes.

David Stidolph: The developers of COMAL 2.0 did not want their language to fall to the side of the road as computing continued to advance, so they put the ability to update COMAL by adding machine language "packages" to programs, and the

Len Lindsay: I don't know assembly language - and I don't want to! Yet I had to read this book before we published it. And you know what? It reads so well, I was almost tempted to learn it. Jesse did a marvelous job of relating dry facts and details in a very readable way. This book is the key to creating your own packages for use with the COMAL 2.0 Cartridge. I can't wait for the sequel!

Geoffrey Turney: If you're going to learn a language, why not learn it from the person who created it. That's what this book is. Borge Christensen, founder of COMAL, introduces COMAL in a fun, humorous way with good exercises and attention getters randomly placed throughout the book. Just in passing - many of the names and people in the book are REAL people (some were upset about being in the book).

Len Lindsay: This book has become a favorite. It is so popular that it is out of stock most of the time. We just got 500 copies in a couple months ago and they were gone within weeks. The problem is that this book must be imported from England and we have problems getting it (it seems that we are by far the largest purchaser of the book). The publisher maintains such a low inventory that it must be reprinted nearly each time we order more! If you are a beginner, I think it would be hard to go wrong with this book - just be patient when you order it.

John Main: A computer language can seem extremely complicated to a young student. Computers are supposed to be fun! If results come only after 3 hours of coding, attention wanders and initial enthusiasm wanes. Most computer texts do nothing to thwart this decay of interest. They are lifeless and dull, intended only to supplement a class curriculum. I remember a computer class where the text had no practical examples at all, unless you were interested in the curvature of a hanging cable, and didn't even suggest touching the computer until chapter 5.

Happily BEGINNING COMAL has avoided these problems. As a leading educator in Denmark, Borge Christensen has successfully written a hands on COMAL tutorial aimed at the beginning computer user. Assuming you have had no previous computer experience, this book will teach you to program in COMAL.

A wonderfully direct technique is used to reveal the power and beauty of COMAL. Chapter 1 begins with this program line:

```
25 PRINT "HI THERE."
```

By dissecting this simple line, line numbers, statements, keywords, and string constants are introduced. While still on page 1, the student is already presented with a hands on example to run. By building on these short simple concepts with a complete series of examples and exercises, a student is lead from "print your name" through variables,

conditionals, iteratives, and into file structures. As each new concept appears, a clear example of its usage is given, along with exercises to show why it works, and the commands to implement it.

The only problems in this book are the chapter titles. After reading the book and working the problems, I wanted to look back at a concept. I first had to remember the keyword and look it up in the index. A chapter heading such as "Little John Takes A Note Home" may be cute, but does little to help the reader realize that the keywords ZONE and SELECT are discussed in the chapter.

Some of the examples in BEGINNING COMAL have a distinctly European flavor (Mettwurst?), but for the most part this book is very well written, clearly developing the topics under discussion. While another book might be preferred as a reference for the serious programmers library, BEGINNING COMAL should be your text of choice for teaching COMAL or any beginning programming language. [from COMAL TODAY #2]

David Skinner: This text contains 160 pages of text and another 150 pages of program listings, self tests, and answers to exercises. This particular book is a programmed instruction course in the classical sense. There are blanks for you to write in your own answers throughout the book. There are structure diagrams galore, to represent each principle being considered. This is one book that you cannot use without the computer. While the COMAL Users Group did include an errata sheet with the book, it was really more an instruction sheet for using the accompanying disk; there were very few errors of any kind. The book begins with 'the computer writes a message' and before it ends we have covered data management, accounting, statistics, etc. in a somewhat superficial way. Be assured I do not mean that in a negative way, the text remains light and flows gently from one subject to the next. If I had to recommend a best first book for beginning

COMAL, I would have to recommend, BEGINNING COMAL by Borge Christensen. [from Clark County Commodore Computer COMAL Club]

## CAPTAIN COMAL'S GRAPHICS PRIMER MINDY SKELTON

84 PAGES -- \$19.95 / \$12.95

Geoffrey Turney: I had no idea that Mindy was interested in graphics. But for me, who knows nothing about it, let alone sprites, her book pointed everything out - simply and easily. I actually managed to put up a graphic character and move it, using this book (Hint - I copied some of the programs from the book). I even found out that the turtle has a HEAD (see page 20). Now I understand what this big commotion is about - graphics and the turtle and little children liking it! I was always under the impression that MIT's turtle was in actual fact a HERO robot. But now, I found out that it's name is CALVIN the COMAL TURTLE. An excellent book for those not interested in graphics but who know that they still need to 'get into it' for future CAD/CAM type systems.

Len Lindsay: GRAPHICS PRIMER provides a nice introduction to turtle graphics and sprites. It makes enjoyable reading including illustrations. My favorite was a comparison between a real turtle and a COMAL turtle.

Jim Ventola: Mindy Skelton's book is number five in "The Amazing Adventures of Captain COMAL" series, and looks much as the others do. It is a pamphlet really, having been produced on Easy Script, printed on a dot matrix printer, and then reproduced and published in a paper cover without a binding (except for two staples). It is attractive enough. However, since right justification was used, the page is littered with extra spaces between words, which I for one

Anyhow, what matters is the substance of the book. It is, indeed, a primer in the sense of "an elementary textbook"; it assumes no previous knowledge of COMAL or COMAL graphics. After a discussion of COMAL procedures and functions and advice on how to use the disk intelligently, Ms. Skelton leads us through Turtle graphics and Sprite graphics. In addition, she provides several ways to create sprites, a library of useful procedures and functions, a glossary and an index. Moreover, there are many example listings (provided also on the accompanying disk) for study.

David Stidolph: Three different textbooks have been published to help people learn COMAL. However, none of these textbooks teach anything about COMAL's graphics or sprites. Now, thanks to Mindy Skelton, COMALites can learn about the graphics abilities of their computer. The book, over 80 pages in length, is a good introduction to drawing graphics and sprite control.

Sprite control is one of the hardest graphics features to learn, but Mindy's book comes to the rescue with step-by-step instructions on what a sprite is and how to create and move it. Not one or two, but three ways of creating sprites is shown.

Since the book comes with a matching disk that contains all the programs from the book typed in, very little typing is needed to try the examples. The disk has a menu of the sample programs, so you just LOAD and RUN that program, and you can select any of the examples to RUN.

**64 PAGES -- \$9.95 / \$6.95**

David Stidolph: COMAL 2.0 on cartridge for the Commodore 64 is the most powerfull 8-bit language available. All the machine dependent features (sound,



Len Lindsay: If you use the COMAL 2.0 Cartridge, this book is essential. That may be why it is part of the DELUXE COMAL CARTRIDGE package set. While not as big and detailed as the COMAL HANDBOOK, it is a good companion, providing a reference to all the new commands of the eleven packages built into the cartridge.

The book reads well, and in fact, curled up by the fireside, it's done before you know it. (I'm still hoping for a sequel!) However, beyond the reading, it is an indispensable reference book for future cartridge programming.

Geoffrey Turney: Len has done it again. He has organized graphic routines and keywords into an easy to look up reference book with commands that were omitted from his COMAL HANDBOOK. It has alphabetical / categorized sample programs explaining the keywords with additional samples and further defined keywords to work on. You just cannot get

David Stidolph: This reference manual on graphics for COMAL 0.14 is quite good. Every graphics and sprite command has at least one page to itself, with the average better than two pages, including a sample program. This book is a good reference manual, but is NOT a tutorial. One advantage is it avoids the constant page-flipping to look up tables in the back of the book. When a table is needed (like the colors and their numerical values) it is printed on that page.

Jim Ventola: COMAL 0.14 is a great language for doing graphics on the 64 -- if you have documentation, time, patience, and something going on in your brain's right hemisphere. When I first got into COMAL (in the summer of '84), the only documentation I had was the first edition of the COMAL HANDBOOK and a list of graphics commands on a sheet of paper. Luckily, my COMAL SAMPLER disk had some demonstration programs that I could ponder, study, and take apart. But now, the student of COMAL graphics has a handbook to explain the whole system.

ISSUE #7 - COMAL TODAY, 5501 GROVELAND TERRACE, MADISON, WI 53716 - PAGE 39



offering a diskette with all the programs for \$19.95 plus tax, but I preferred to type in the programs myself. And since the explanations of the keywords are accurate but laconic, you really will need to study some of these programs carefully.)

The book is addressed to the beginning COMAL user, so it can be used by just about anyone. After an overview of COMAL (worth reading even if you are familiar with COMAL), the subject is treated in three main categories: sprites, turtle graphics and graphics. There are also five appendices and an index.

The book is nicely produced by Reston, with generous margins, good binding, and attractive typesetting. Since I found it at my local bookseller, I imagine that Reston hopes to compete with the plethora of guides to SID and VIC II written for BASIC programmers. I don't know whether the general public will support a book that requires sending off for a disk to get the language that the book teaches. But I do know that anyone who already owns COMAL for the C-64 will want to own this book.



## FOUNDATIONS IN COMPUTER STUDIES

WITH COMAL

JOHN KELLY

363 PAGES -- \$19.95



Geoffrey Turney: Everyone has to start somewhere. I like to start at the beginning. FOUNDATIONS starts at the beginning and brought me to the point where I can say that I am COMAL COMPUTER LITERATE. It is an amusing, well written book that allows you to understand the difference between data files and regular files. It also explains what bigger computers are used for and how they assimilate, digest, process and spit out their data.

David Stidolph: FOUNDATIONS is the textbook that I would suggest a school use, because it covers not only the language, but gives an overview of computers as well. It was written for COMAL on the Apple II, but it covers C64 COMAL quite well. There are some small differences, but a page of notes comes with the book to explain them. Each chapter has a "Summary" and "Questions and Exercises" sections that cover the chapter well. This book is in its second edition, and has added a chapter for graphics (using the Commodore 64 version) and updated references and pictures.

Len Lindsay: Here is a book imported from Ireland that not only presents the fundamentals of COMAL, but provides a foundation for studies about computers and how to use them (thus its title is long but very appropriate). The question of which tutorial book is the best may never have a answer, but this book has quite a following. It is used by school systems in Ireland. While I have met the author of nearly every COMAL book, I still look forward to the privilege of meeting John Kelly some day soon.

David Skinner & Clay Ratliff: This is the most difficult review of all because one of us likes the book and the other (Clay, 10th Grade) has rejected the book (probably for budgetary reasons). This textbook is a good value, 363 pages of solid information. It seems to have been written for an APPLE Computer but since it is in COMAL, we really could not tell the difference. Over 100 sample programs are on the disk which (optionally) accompany the book and the one page of notes from the COMAL Users Group explains the reason for any differences in the programs. The book starts out as a hands on tutorial, teaching you how to write and save simple COMAL programs. From there, the book goes into the theoretical aspects of structured programming with immediate applications in COMAL. It was Clay that first noted that the book was teaching theory, I thought it was just being interesting. The book also digressed to a facinating study of the

variety and history of the computer. Clay felt the highlight of these chapters was the 1946 picture of a 30 ton Computer. What I found even more facinating was the books treatment of multidimensional arrays, random and direct access files, and recursive routines. The author concludes with brief chapters regarding the applications of computers and the social and ethical implications of computer dataprocessing.

Overall I found this book to be most enjoyable and informative, not only with specific COMAL applications but also in establishing a good foundation. Clay felt that the book was just too much like a Computer Science textbook for him to want at the end of the school year. [from Clark County Commodore Computer COMAL Club]

## \*\*\*\*\*

### CAPTAIN COMAL GETS ORGANIZED

LEN LINDSAY

102 PAGES -- \$19.95 / \$12.95

\*\*\*\*\*

Geoffrey Turney: This was the first book I read in the series of Captain COMAL books. It addressed my problem of having far too many disks and not being able to access a particular program without doing a DIRectory of many disks to find it. At the same time - it organized my thoughts on how to use COMAL. It has lots of little modules which can be merged into other programs and used - the wonder of COMAL.

David Stidolph: This is the first book of the famous Captain COMAL series. It shows a simple way to write disk catalog programs by breaking the large task into small parts. The book explains the concept of procedures and functions, and chaining between programs. As a learning tool this book is excellent. The disk contains the programs from the book already typed in and ready to run. The idea of a disk catalog program that can

locate any program on any disk is a good one, but the program lacks true power. It takes too long to read in a disk directory, and in order to search for a program you have to know its name. No ability to search, or list a "category" of programs is allowed. Perhaps the ability will be added in a future version.

Len Lindsay: COMAL has many advantages over BASIC. But someone used to BASIC may not discover this COMAL 'style' easily. This book is designed to constantly emphasize and explain procedures and functions. When you are done with the book, you will understand how important they are and how to use them to make writing your own programs much easier (some programs at the end of the book were over 2/3rds done before they were even started, just by using modules already written for previous programs). That is the goal of the book. The fact that you also end up with a simple disk management system is only a side benefit.

David Skinner: I had promised a review of CAPTAIN COMAL GETS ORGANIZED and here it is. By now most club members have seen the program in action. The programs are a Disk Management System (DMS) which will help prevent using the same disk ID more than once and maintain a catalog of all programs on all disks. I found that most club members with only four to ten disks would not be interested in this program. When you buy your second box of disks you will need to get organized. I spent an afternoon recording the directories of more than fifty disks on the master directory. It takes less than one minute to search every directory on every disk for the program that you want. Since it is time consuming to switch from Word Pro to COMAL and back again, I really appreciated the directory printout routine. We were able to printout the directories of every disk in a multicolumn format on continuous form index cards. Updates are very easy and the master file is resorted only when you want to resort it. All in all, this disk

has to be the best value for your dollar I have ever seen [from Clark County Commodore Computer COMAL Club with permission].

David Skinner: Last issue, when I wrote my first review of CAPTAIN COMAL GETS ORGANIZED, it was based upon a prepublication copy of the disk. The program worked so well that I gave it the best review possible. Now that I have the final version in my hands and am able to read the book, I discover that I had completely missed the point. This book is an applications tutorial which can take someone who has never used a computer and teach them to design and write a data management program, 'the example in the book was the Disk Management System. When I first read the book, I really was not that impressed. Because of the modular form used to write the programs, I had already decided that making some minor changes to the program would be very easy to do, even without the book. My wife, however, thought the book was wonderful, informative and entertaining. I really did not appreciate Lindsay's talent as an educator until we tried to use this book as a COMAL teaching aid.

Dr. Philip Clark was our test pupil. He started at the beginning, doing the exercises and in effect holding a conversation with the book. He made every mistake that the author expected a reader to make, and he was successful in writing his programs. The strange thing is that when he was finished, his Disk Management SYSTEM seemed not to work as I would have expected. He has ended up with a CLIENT MEMORANDUM SYSTEM. This little book is incredible! If you have a lot of disks, you will want the DMS; if you are just starting out, this book makes it very easy. Having gone through the book a little more carefully, the ONLY fault I can find with it is its failure to consider RANDOM files. Perhaps the author will do a sequel soon which will cover more advanced file handling method. I am quite certain it will be well received. [from Clark County Commodore Computer COMAL Club]

## STRUCTURED PROGRAMMING WITH COMAL ROY ATHERTON

266 PAGES -- \$26.95

Geoffrey Turney: If you already know a language or two, and just want to find out about how COMAL works. this book is for you. It is written in a serious, straight forward tone, and I got stalled on page 30.

David Stidolph: This textbook by Roy Atherton is what the title says it is (how's that for truth in advertising?). One of the advantages of this book is that it shows how to write a program by defining the problem and breaking it up into manageable sections. Also, it covers advanced topics like sorting (3 different types) and search keys in file handling. Since the book was written for COMAL in general, small differences exist for the Commodore version, but the reader should have little problem dealing with them (a page of notes comes with the book).

Len Lindsay: I learned programming on my own, buying many tutorial books over the years, as well as subscribing to nearly every computer magazine. Since I lacked formal programming training, this book showed me quite a few things that I hadn't known. I enjoyed the practical style of the book and think it would be a fine book to learn programming techniques and styles. It is imported from England.

David Skinner: This review is mixed. We have been reading STRUCTURED PROGRAMMING WITH COMAL. We were fortunate to have ordered the book from the COMAL Users Group as the ERRATA sheet that they provide was most useful. When you first read this book, start with the cover. The boys on the cover are playing rugby (football in England). Some of the explanations in the book are so simple that any schoolboy in England could readily understand it. If you have had no exposure to English culture, some of

This book stresses the concept of a structured program, something which the novice can readily grasp and learn without delay but may leave an experienced programmer shaken and sweating. If you are an experienced programmer, your best course of action is to begin at the beginning of the book, perform every exercise, note every time you disagree with the author and why, don't skip forward, and when you are finished you will be a better programmer and perhaps a better person. While you may want one copy of the COMAL HANDBOOK near the machine for reference, you will more likely be needing one copy of STRUCTURED PROGRAMMING WITH COMAL by Roy Atherton for every member of the household interested in computers. Sit down as a family and study this book together, it is certainly more fun than watching reruns of \*\*\*. [from Clark County Commodore Computer COMAL Club]

**320 PAGES -- \$15.00**

This tutorial is written specifically for the C64 COMAL 2.0 Cartridge. Thus it goes one step further than the other COMAL tutorial books - it not only introduces COMAL but presents the graphics and sound packages in an easy to learn manner as well, complete with examples. The book also explains the other built in packages, such as SPRITES, FONT, SYSTEM, JOYSTICK, PADDLES, and LIGHTPEN. Plus the final chapter briefly explains how COMAL and Machine Language can interact. The book includes drawings and illustrations and is written in an easy to read manner. The authors took great care to be as accurate as possible. It was written on a word processor and printed on a near letter quality printer. Currently they are transmitting the complete text electronically to a typesetting system so that the next printing will be typeset.

ISSUE #7 - COMAL TODAY, 5501 GROVELAND TERRACE, MADISON, WI 53716 - PAGE 43

we can highly recommend this book for new Cartridge owners looking for a tutorial. Although it was written in Denmark, one of the two authors is an American, so the English is very good. This is an official Commodore publication, made in Denmark.

\*\*\*\*\*

## COMAL HANDBOOK

LEN LINDSAY

479 PAGES -- \$18.95

\*\*\*\*\*

Geoffrey Turney: The HANDBOOK - an absolute necessity. Any further comments would make this review redundant.

Len Lindsay: I wrote the COMAL HANDBOOK in co-operation with UniCOMAL, authors of Commodore COMAL. It is designed to be a detailed reference to both current versions, 0.14 and 2.0. Since RESTON (the publisher) is rumored to be cancelling all its computer books, you may wish to RUN to your nearest book store and get a copy before it's gone. We have 1000 copies left here, but they will go fast too. Meanwhile, I am searching for a new publisher and hope to minimize the transistion period.

Diarmuid McCarthy: This book is more a manual or reference book than a textbook. It is, however, essential for anyone who wishes to learn COMAL on Commodore computers [including C64]. The main part of the book is in the form of a reference manual covering all the COMAL keywords. For each keyword there is an explanation, its syntax is given, and there are examples and sample programs. Whether or not each keyword matches the standard is stated, as are the versions in which each keyword is available. There are many appendices, some of which are very useful. There are special sections for the COMAL structures, string handling, and useful procedures and functions. The COMAL KERNAL is also given.

For someone who already knows PET BASIC and its operating environment, this book should be extremely helpful to them in learning COMAL on a Commodore computer. Some users may however require additional assistance in getting to grips with the more complex aspects of COMAL such as procedures and functions with parameters. The book is not particularly suitable on its own for someone learning COMAL as a first language. It is however, an essential reference book for all who use COMAL on Commodore computers. [used with permission from Riomhiris na Scol, Colaiste an Spioraid Naoimh, Bishopstown, Cork, Ireland]

John Main: So you've heard about COMAL and you're ready to switch. The newest and best of the high level programming languages is about to make your programs fast, efficient, and fun to write. And why not? It is inexpensive, easy to use, and extremely powerful. But you need a reference book to show you the way.

Let the COMAL HANDBOOK be your guide. It contains the most complete description of the COMAL language to be found anywhere. And I do mean COMPLETE! Within the book's 479 pages every COMAL keyword is listed on a separate page, well defined at length, and shown in at least one USEFUL program sample. Each page continues with extensive cross-referencing to both related keywords and other example programs which contain the keyword.

The Keywords in the book are in alphabetical order, each on a separate page. This allows ample room for an in-depth discussion of how, when, and where it should be used, including which versions support which keywords. The standard syntax is listed first, with default values and possible ranges for each value in a clear and easy to understand format. Next comes one or more examples of how the keyword looks in a working program, with both user inputs and computer responses shown. Finally, Additional Samples shows where to find other examples, Used in Procedure

indicates which of the procedures at the end of the book contain this keyword, and See Also shows a list of related keywords.

The appendices contain a complete guide to COMAL structures grouped to show their relationship to each other. Each structure is then separately defined, with examples, to show how to use it in a program. Appendix B shows how to convert Basic string handling to the much faster COMAL forms. Sequential file differences are discussed in Appendix C, because COMAL allows you to write sequential files in two different ways. A LONG list of practical procedures is included in Appendix D to help you build your own COMAL procedure library. A complete listing of the COMAL Kernal, the definition of Standardized COMAL, is also included, and naturally, a complete index ends the book.

You have found your guide to COMAL. Together with this book, you have the most efficient programming language working for you. It's as if the authors of COMAL are standing behind you. The answer to your question has already been answered. Complete is the only word to describe the COMAL Handbook. When you need to know something about COMAL, this is the place to look. [reprint from COMAL TODAY #2]

Bob Schulz: The COMAL HANDBOOK was the ideal book to help me learn COMAL. I had past experience in Pascal, Fortran, Forth and Basic and wanted to learn COMAL as fast as possible. I felt that the introduction to the book included just the right amount of background information. The command words are listed in alphabetical order, in a structured manner, with an example for each word. Hence, any information I wanted could be found rapidly.

My method of learning COMAL then became one of reading the definition of a command word and the example. If I didn't understand the example, I looked up the command words that I had questions about.

(If I had a Commodore 64 around I probably would have typed in the examples and modified them to see the effects.)

I still keep my dog-eared copy of the COMAL Handbook nearby for reference when I am writing COMAL Programs. I wish all handbooks and manuals were as well written and organized as the COMAL Handbook.

Donald Dalley: In the Commodore world, there are few examples of legitimate copies of any program spreading faster and more widely than that of the COMAL 0.14 language. Being freely copiable, most user group libraries include the original "COMAL SAMPLER" disk for the C64. This successful ploy teased thousands into at least giving COMAL a chance. Within ONE year, an unprecedented number of public domain programs were in circulation!

The first edition of the COMAL HANDBOOK, covering the earliest versions of the language, was the main reference guide for these ambitious programmers. This edition was helpful, informative, and necessary to read, but new versions of COMAL required an updated edition.

Two major differences stand out between the two editions. The second edition has about a third more pages and the spine is perfect-bound, instead of plastic spiral bound. Two problems arise from this change: 1) the book will not lie flat and will need controlling, usually with one of the typing hands; 2) glue holding pages together breaks down, eventually leaving you with a number of loose pages.

The main body of the HANDBOOK is dedicated to the description of over 140 keywords arranged alphabetically. Each word has its own page(s) devoted to explain its use, with the keyword in bold print in each corner for easy page flipping. Whether the keyword is part of the Kernal or not is specified. This is handy for those who wish to write programs for other computers. How fully



the keyword is implemented, if at all, is indicated in simple code form (-,+,\*) for both versions. This is useful for those with the 2.0 cartridge who wish to write programs for the 0.14 disk version. A clear, concise description of the rules for the keyword's use follows, along with notes usually naming limitations or restrictions.

You can't write programs knowing only keywords and no structures, so the importance of the expanded appendices can't be understated. The language's structures are explained with the same precise detail as in the previous section. More useful program listings are included, along with over 130 built-in COMAL 2.0 error messages (BASIC V2 has 28 plus any from peripherals), the KERNAL's definition, trouble shooting hints, explanation of machine language LINKage, disk file usage, disk commands, and other very important things. Some topics are worthy of their own separate books.

The last part of the book is the index, which looks much like the CONTENTS page with a few additions; again, useful. Not covered are the "extra" packages in the cartridge (Turtle, graphic, sound and others) and an important omission TRACE.

The importance of the book lies in its design and usability. You can find what you want, when you need it. If you make a mistake, you can find out why your idea was not allowed. No extra words get in the way of what needs to be explained. Len writes the way manual authors should. Like COMAL itself, the COMAL HANDBOOK second edition is exemplary. [with permission from TPUG Magazine]

## SUMMARY OF BOOKS

Geoffrey Turney: The COMAL HANDBOOK and COMMODORE 64 GRAPHICS WITH COMAL are absolutely necessary to properly use the Commodore COMAL. Depending upon how advanced you are with computers, I would say that absolutely new beginners should read through FOUNDATIONS IN COMPUTER STUDIES WITH COMAL; people with some idea of what their computers can do (consider themselves INTERMEDIATE) would like BEGINNING COMAL. Advanced computer users who also use other languages: STRUCTURED PROGRAMMING WITH COMAL would satisfy their need for INFO on COMAL. The CAPTAIN COMAL books supplement the books just mentioned with added examples and more specific information.

Len Lindsay: I think we are fortunate in that all 13 COMAL books are very good -- you should be happy with any of them. However, keep in mind that the tutorial books are designed to help you learn COMAL and can't really be used as a reference. Likewise the reference books are just that, and not designed to teach you COMAL step by step. If you are new to programming, the COMAL WORKBOOK and TUTORIAL DISK are a good way to get started. Starting out with the cartridge you should use CARTRIDGE TUTORIAL BINDER. For reference, the COMAL HANDBOOK is the detailed book for both versions; COMAL FROM A TO Z is less expensive but condensed for version 0.14.

David Stidolph: How do you summarize 13 books in one paragraph and make recommendations to people you don't know? You don't. The quality of the books deserve more than one sentence, but that is all the space there is. The way to choose which books are for you is to look at where you are now (in programming competence), how far you want to go, and to find the book that fits that description. It's hard to go wrong on any of the books, but try to concentrate on an area of interest (graphics, sprites, program development, ect.) and find a book that covers it.



## BEST SELLING COMAL BOOKS

Last issue we mentioned three NEW COMAL books that we thought would make it into the top 5. They all did. COMAL WORKBOOK all the way to #1, COMAL 2.0 PACKAGES up to #2, and COMMODORE 64 GRAPHICS WITH COMAL as #5. There are no new COMAL books to announce this issue, but 3 or 4 new manuscripts are in the works. A common question is "What book should I get?". To answer that question, we are printing reviews of all the COMAL books in this issue. Hopefully that information will be useful to all our readers. If you would like to meet three of the authors of books up in the top 2, just attend the MARCA COMMODORE SHOW the last weekend of July (see the ad in this issue). And just in passing - the COMAL 0.14 CHEATSHEET keyboard overlay is becoming extremely popular. If it were a book it would be in the top 5.

### MARCH 1985

- #1a- COMAL HANDBOOK  
by Len Lindsay
- #1b- BEGINNING COMAL  
by Borge Christensen
- #2 - CARTRIDGE GRAPHICS AND SOUND  
by Captain COMAL's Friends
- #3 - COMAL FROM A TO Z  
by Borge Christensen
- #4 - FOUNDATIONS IN COMPUTER STUDIES  
by John Kelly
- #5 - COMMODORE 64 GRAPHICS WITH COMAL  
by Len Lindsay

### APRIL 1985

- #1 - BEGINNING COMAL  
by Borge Christensen
- #2 - COMAL 2.0 PACKAGES  
by Jesse Knight
- #3 - COMAL HANDBOOK  
by Len Lindsay
- #4 - CARTRIDGE GRAPHICS AND SOUND  
by Captain COMAL's Friends
- #5 - COMAL FROM A TO Z  
by Borge Christensen

MAY 1985 - (first half of month)  
(Final list next issue)

- #1 - COMAL WORKBOOK  
by Gordon Shigley
- #2 - COMAL 2.0 Packages  
by Jesse Knight
- #3 - COMAL FROM A TO Z  
by Borge Christensen
- #4 - COMAL HANDBOOK  
by Len Lindsay
- #5 - CARTRIDGE GRAPHICS AND SOUND  
by Captain COMAL's Friends

## MORE ON COMAL SUBSTRINGS

Last issue, page 73 showed you how easy substrings can be with COMAL. But David Tamkin reminds us to be careful. For example, look at this:

```
DIM test$ OF 10
test$:="a"
test$(2:3):="bc"
test$(5:6):="ef"
```

We have allocated a maximum of 10 characters for TEST\$. Then we assign TEST\$ to be equal to "a". Thus TEST\$ is one character long. Now we assign "bc" to be the substring from character 2 thru 3. This is OK because it is contiguous with the existing TEST\$. But the final line will yield an error. Since a fourth character doesn't exist in the current TEST\$, COMAL cannot tack on a fifth and sixth character! Thus when assigning substrings, make sure it falls within the current string, or at least starts with the next character.

This same idea applies to reading a substring. An error results if you try to read a substring that goes past the last character of the current value of the string. Thus the following line would be in error if it replaced the last line of the previous example:

```
PRINT test$(8:9)
```

# Interstellar Dust Clouds

by Tom Kuiper 1985

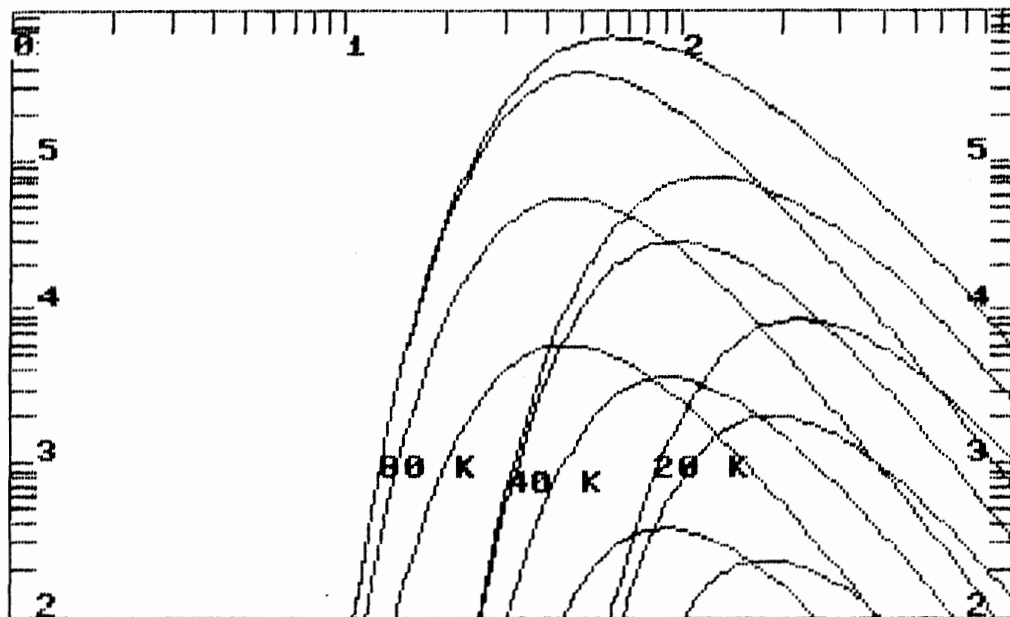
The COMAL 0.14 data graphing package presented on page 40 of COMAL TODAY 5 has been converted to COMAL 2.0 and extended to include logarithmic axes. It is included as LST.GRAPH5 on the accompanying disk. The original 0.14 version has a minor correction and is included as GRAPH52.L.

In order to illustrate the 2.0 version, I've also included a 2.0 program called CLOUD'FLUX.LST. This plots the intensity of emission from interstellar dust clouds for infrared wavelengths from one millionth of a meter (one micron) to one thousandth (one millimeter = 1000 microns). The numbers along the logarithmic axes are exponents. Thus, 2 means "10 raised to the power 2" or 100. Be patient at the beginning; since the first values computed are outside the range of the screen, nothing will appear

to happen for a while. (For those who would like to know a little more about the program, an explanation follows.) After the graph has been drawn, you are asked if you want a copy. You will have to insert an appropriate screen dump routine after line 530. You can try changing cloud temperatures (line 50) and wavelengths (line 160). For example, visible light ranges from 0.4 micron (deep violet) to 0.77 micron (dark red). The InfraRed Astronomy Satellite was sensitive to wavelengths between 6 and 120 microns. IRAS delighted the world with spectacular new views of the Universe from January 25, 1983, when it was launched, until November 21, 1983, when it ran out of liquid helium and ceased operation.

EXPLANATION: A solid body radiates and absorbs energy at all frequencies of the electromagnetic spectrum. (Line 300 in the program converts wavelength in microns

More ▀



to frequency in billions of a cycle per second, or gigaHertz.) The ability to absorb or emit radiation varies with frequency according to the properties of the surface. Colors are a consequence of this. Physicists speak of a "blackbody" as something which is able to absorb or emit radiation with complete efficiency at all frequencies, that is, one which reflects none of the radiation and therefore has no color. The actual amount of radiation emitted by a blackbody depends on its temperature. The hotter the object, the higher is the frequency where most of the energy is concentrated. On December 14, 1900, Max Planck presented the law which describes this behaviour (and now bears his name) to the German Physical Society. At the time, no one was aware that this paper, coming two weeks before the beginning of the new century, would initiate a chain of discoveries which would overturn physics, lead to the atom bomb, the laser, computers, and COMAL. FUNC BB'BRIGHTNESS(T,GHZ) calculates the brightness of a blackbody of temperature T (in degrees Kelvin) at a frequency GHZ according to Planck's Law.

Dust in interstellar space consists of very tiny particles. Although astronomers speak of "dust grains" the particles are more like smoke than sand. In fact, the particles are small compared to the wavelength of visible light. Such tiny particles cannot absorb or emit long wavelength radiation very well. The longer the wavelength, the worse it gets. This effect is taken into account in line 320 which computes the opacity (opaque-ness) of the clouds. The actual amount of dust present is also a factor determining whether a cloud is opaque (and therefore visible) or transparent at a given wavelength, as seen in line 320. ■

## INPUT ON INPUT

by Bill Wood

I have discovered a very handy COMAL capability. You can stop a program during an INPUT request, execute procedures from direct mode (even including other INPUT requests), and still return to the original via the CON command. Type in the program listed below (remember to use AUTO):

```
INPUT "Enter a number: ": a
//
PROC demo'a
  PRINT "See what we can do!"
ENDPROC demo'a
//
PROC demo'b
  INPUT "Enter Demo'B number: ": b
  PRINT b
ENDPROC demo'b
//
PRINT a
END
```

Now try these three things:

- 1) Normal run:
  - \* RUN program
  - \* Type 33 as your reply
  - \* Program ends
- 2) Execute a PROC and return to INPUT:
  - \* RUN program
  - \* STOP program with STOP key
  - \* Type: EXEC DEMO'A
  - \* Type: CON
  - \* Type 33 as your reply
  - \* Program ends
- 3) Get another INPUT and return to first
  - \* RUN program
  - \* STOP program with STOP key
  - \* Type: EXEC DEMO'B
  - \* Type 99 as your reply
  - \* Type: CON
  - \* Type 33 as your reply
  - \* Program ends

## A PROBLEM WITH PACKAGES

by Jesse Knight

COMAL 2.0 has the ability to link information to a program. This linked information is saved and loaded with the program, automatically and can be sprite images, ML packages, and fonts. This is something you probably knew already.

What you may not know is that they all don't work TOGETHER. To be more specific, a program with one or more packages AND either sprites or a font linked to it will not LOAD properly. They can be linked, and will work fine, with a program in memory. This is true of both the EPROM and ROM versions of the cartridge.

This is caused by an undesirable "feature", in other words, a bug, in the LOAD routine. It prematurely closes the program file after loading the program and package(s), but before loading the font or sprites. This causes future calls to the Kernal routine CHRIN, which is used to read the data from the file, to default to the keyboard.

For keyboard input, CHRIN turns on the cursor. This makes it appear the load finished properly. When a command is typed in, followed by return, the computer will ignore it. It thinks it is reading data from a file, not the keyboard. To regain control, press the RUN/STOP and RESTORE keys.

Now the computer will respond to commands. There's just one problem. Neither the font nor the sprite images have been loaded.

The moral of that story is simple. Don't try linking packages and either a font or sprites to a program, at least until someone discovers a way to solve this problem.

## COMAL 0.14 ERROR MESSAGES

by Dick Klingens, Dutch COMAL80 Group

COMAL TODAY #3, page 7, describes the access to the error messages. However, the PROC given in the article can not be used frequently in a program because the procedure call aborts the program. Instead of the described procedure I suggest:

```
PROC ERROR'MESSAGE(ERROR) CLOSED
  POKE 45,ERROR
  SYS 6472
  PRINT
ENDPROC ERROR'MESSAGE
// demonstration program:
REPEAT
  INPUT "ERROR NUMBER: ": NUM
  ERROR'MESSAGE(NUM)
UNTIL NUM=0
```

By the way, it is possible to add error texts to the file COMALERRORS using this program:

```
DIM TEXT$ OF 40
TEXT$:="END OF EXECUTION"
ERRORNUM:=105
SEV:=0
LNG:=LEN(TEXT$)
OPEN FILE 9,"COMALERRORS",APPEND
PRINT FILE 9: CHR$(ERRORNUM),CHR$(LNG),C
HR$(SEV),TEXT$, // wrap line
CLOSE FILE 9
```

I think the program GENERATE ERRFILE must be changed in the same way, using only error number, severity and error string in the DATA statements and replacing the reading and writing by:

```
READ NUM,SEV,TEXT$
PRINT FILE 9: CHR$(NUM),CHR$(LEN(TEXT$)
),CHR$(SEV),TEXT$, // wrap line
```

because then you can forget the count of the number of characters in the error string preventing mistakes (miscounts).

## 5 BYTE INTEGER - NOT 2

Several readers have discovered that COMAL 2.0 stores an integer as 5 bytes when writing it to a file. D.S. from Ontario investigated the issue, complete with track and sector displays of integer files. Here is part of his comments:

I ran into one problem that I didn't expect, and that is the file incompatibility when integers are written to a SEQ file. It seems COMAL 0.14 converts integers to floating point format for disk storage, and converts them back when read back into an integer variable.

The evidence for this is in the SEQ files on my disk. The contents of these files are printed out below. Contrary to Appendix C of the COMAL Handbook, the WRITE command in COMAL 0.14 stores integers the same way it stores floating point numbers, in 5 bytes. COMAL 2.0 expects to read a 2 byte integer, and gets upset. I suppose I could work around the file incompatibility by reading the data back into a floating point number, but I think it would be a good thing if you let people know what the problem is, so they won't waste too much time beating their heads against the wall.

track 17 sector 2

**TEST'FILE 0.14**

```
00 :00 19 00 0c 53 43 48 45 4c 4c 45 4e 42 45 52 47 : schellenberg
10 :84 45 85 1e b8 8d 54 28 00 00 0c 93 c5 00 1e 25 :DeE iMt( SE %
20 :b5 03 1a c5 4e 54 45 52 20 41 20 53 54 52 49 4e :f Enter a strin
30 :47 20 28 4c 45 4e 3c 3d 31 32 29 20 3a b7 09 02 :g (len<=12) :u
40 :ba bb 00 28 20 b5 03 15 c5 4e 54 45 52 20 41 4e :wx (f Enter an
50 :20 46 2d 50 20 4e 55 4d 42 45 52 20 3a b7 07 03 :f-p number :u
60 :b8 bb 00 32 1d b5 03 12 c5 4e 54 45 52 20 41 4e :ix 2f Enter an
70 :20 49 4e 54 45 47 45 52 20 3a b7 08 04 b9 bb 00 :integer :u gx
80 :46 15 03 0f 30 3a 54 45 53 54 27 46 49 4c 45 30 :f 0:test'file0
90 :2e 31 34 9e 00 5a 1b db 02 00 02 dc 03 0f 30 3a :.14n z e u 0:
a0 :54 45 53 54 27 46 49 4c 45 30 2e 31 34 dd df 00 :test'file0.14ef
b0 :6e 13 c3 02 00 02 c8 cb 06 02 c6 04 03 c4 05 04 :n C HK F D
c0 :c4 c7 00 78 08 cf 02 00 02 d0 00 82 05 7d 01 00 :DG x D P B }
d0 :8c 04 00 00 96 04 00 00 a0 09 70 05 83 8a 00 7c :L V / p CJ !
e0 :00 aa 07 02 00 04 46 00 b4 0c 8c 8f 06 00 02 00 : v f t LD
```

track 17 sector 3

**TEST'FILE 2.01**

```
00 :00 16 00 0c 53 43 48 45 4c 4c 45 4e 42 45 52 47 : schellenberg
10 :84 45 85 1e b8 1a 85 49 4e 54 45 52 0c 00 00 00 :DeE i Einter
20 :48 41 52 44 43 4f 50 59 12 00 00 00 53 45 54 52 :hardcopy setr
30 :45 43 4f 52 44 44 45 4c 41 59 0b 00 00 00 53 45 :recorddelay se
40 :54 50 41 47 45 09 00 00 00 49 4e 4b 45 59 08 00 :tpage inkey
50 :00 00 46 52 45 45 1a 00 00 00 4b 45 59 57 4f 52 : free keywor
60 :44 53 27 49 4e 27 55 50 50 45 52 27 43 41 53 45 :ds'in'upper'case
70 :17 00 00 00 4e 41 4d 45 53 27 49 4e 27 55 50 50 : names'in'upp
80 :45 52 27 43 41 53 45 0e 00 00 00 51 55 4f 54 45 :er'case quote
90 :27 4d 4f 44 45 0a 00 00 00 43 55 52 52 4f 57 0a :mode currow
a0 :00 00 00 43 55 52 43 4f 4c 0e 00 00 00 54 45 58 : curcol tex
b0 :54 43 4f 4c 4f 52 53 0a 00 00 00 44 45 46 4b 45 :tcolors defke
c0 :59 0c 00 00 00 53 48 4f 57 4b 45 59 53 08 00 00 :y showkeys
d0 :00 42 45 4c 4c 0a 00 00 00 53 45 52 49 41 4c 0b : bell serial
e0 :00 00 00 53 45 54 54 49 4d 45 0b 00 00 00 47 45 : settime ge
```

## IT'S NOT THE SAME OLD SONG

copyright 1985 George Jones

Baseball teams have mascots.  
States have flags.  
COMALites ought to have a song.  
Right!?!.

George Jones would like to share the following COMAL SONG with us. He hopes it will generate a chuckle or two. Maybe someone will put it to music and we can all do a follow the bouncing cursor number. It is a new program to update a familiar tune - inspired by the chance utterance of a colleague, Mike Lester. (Permission to reprint in COMAL TODAY is hereby granted, provided this notice is included in the reprinted material.)

O' COMAL ye faithful!

To the tune of...(now what do you think!)

REPEAT

song

UNTIL OOB // Out Of Breath

PROC song

O' COMAL ye faithful,  
Structured and recursive,  
No garbage collections to slow down  
your day,  
More powerful than BASIC  
You won't need those ML tricks  
Sprites, Music and the Turtle too,  
Procedures, Functions, Loops that DO  
There's nothing we can't ask of you,  
COMAL's the way!

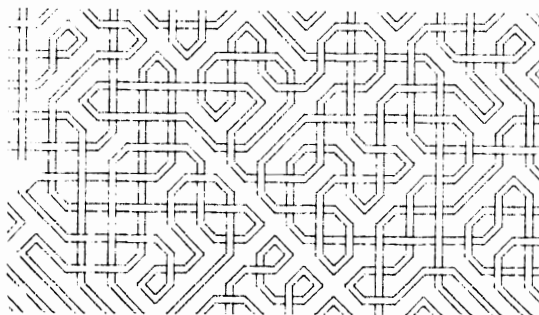
ENDPROC song

OK, it's a bit of an excess but I'll just bet there are a lot more verses out there just waiting to be merged. Don't wait -- send 'em in! I think this is going to be fun.

## FREEWAY

by W. P. Miller

Just a short while ago, Bill M came into the COMAL room on PlayNET to inquire about COMAL. We sent him the BEST OF COMAL disk. The next thing we knew, he had written his very first COMAL program: FREEWAY. He submitted it to the COMALites on PlayNET using Electronic File Transfer. While it was his first COMAL program, it shows a professionalism not found in most peoples first programs. The original program was written in COMAL 0.14, but it was an easy matter to translate it into COMAL 2.0 (add parentheses around graphics statement parameters). The picture following this article was done on a 1520 plotter, and the program that did it is on TODAY DISK #7 (2.0 side, called "1520freeway"). We are printing both 0.14 and 2.0 versions of the program so you may compare to see how they are virtually the same except for the graphics statement modifications.



```
// delete "0:freeway"
// by w.p. miller
// save "0:freeway"
//
init
dummy:=rnd(-256*256*peek(160)+256*peek
(161)+peek(162)) // wrap line
print chr$(147),chr$(14),
print "Generating road map."
print "Takes about 20 seconds..."
for x#:=0 to 340 step 20 do
  for y#:=0 to 180 step 20 do
    c#:=rnd(1,4)
```

More ▀

```

case c# of
when 1
  diagrt
when 2
  upover
when 3
  diaglt
when 4
  upunder
otherwise
  null
endcase
endfor y#
endfor x#
setgraphic
fullscreen
end
//
proc init
  setgraphic 0
  hideturtle
  background 0
  pencolor 1
  border 0
  clear
  settxt
endproc init
//
proc diaglt
  moveto x#+6,y#
  drawto x#,y#+6
  moveto x#+12,y#
  drawto x#,y#+12
  moveto x#+7,y#+19
  drawto x#+19,y#+7
  moveto x#+13,y#+19
  drawto x#+19,y#+13
endproc diaglt
//
proc diagrt
  moveto x#+13,y#
  drawto x#+19,y#+6
  moveto x#+7,y#
  drawto x#+19,y#+12
  moveto x#,y#+7
  drawto x#+12,y#+19
  moveto x#,y#+13
  drawto x#+6,y#+19
endproc diagrt
//
proc upover
  moveto x#+7,y#
  drawto x#+7,y#+19
  moveto x#+12,y#

```

```

drawto x#+12,y#+19
moveto x#,y#+7
drawto x#+7,y#+7
moveto x#+12,y#+7
drawto x#+19,y#+7
moveto x#,y#+12
drawto x#+7,y#+12
moveto x#+12,y#+12
drawto x#+19,y#+12
endproc upover
//
proc upunder
  moveto x#,y#+7
  drawto x#+19,y#+7
  moveto x#,y#+12
  drawto x#+19,y#+12
  moveto x#+7,y#
  drawto x#+7,y#+7
  moveto x#+7,y#+12
  drawto x#+7,y#+19
  moveto x#+12,y#
  drawto x#+12,y#+7
  moveto x#+12,y#+12
  drawto x#+12,y#+19
endproc upunder

```

>>> 2.0 VERSION <<<

```

// delete "freeway"
// save "freeway"
// by w.p.miller
//
init
RANDOMIZE
PAGE
PRINT "Generating road map."
PRINT "Takes about 20 seconds"
FOR x#:=0 TO 340 STEP 20 DO
  FOR y#:=0 TO 180 STEP 20 DO
    c#:=RND(1,4)
    CASE c# OF
      WHEN 1
        diagrt
      WHEN 2
        upover
      WHEN 3
        diaglt
      WHEN 4
        upunder
      OTHERWISE
        NULL
    ENDCASE
  ENDFOR y#
ENDFOR x#

```

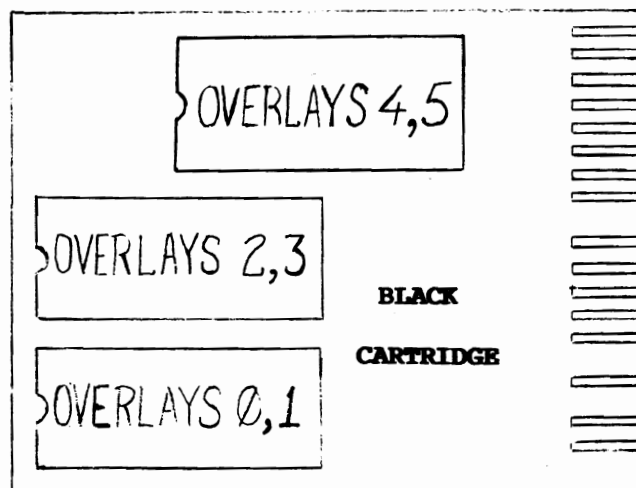
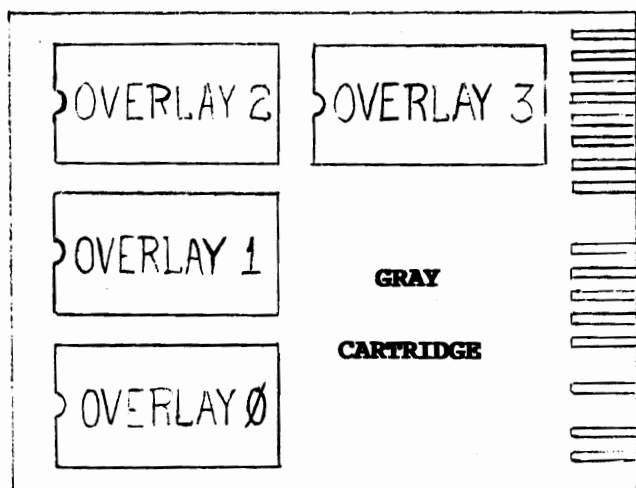
More





## THE CHIPS IN THE CARTRIDGE

C64 COMAL 2.0 is a 64K system. In the gray cartridges, this is stored on four 16K EPROMs. In the black cartridges, it is stored on two 32K ROMs. The COMAL system is organized to bank itself in and out of the computer as needed, in 16K segments or overlays. Thus the 64K system consists of four overlays, named OVERLAY 0, OVERLAY 1, OVERLAY 2, and OVERLAY 3 (computer people like to start counting with 0). Bob Schulz drew the diagrams below showing where these overlays are. Note that the black cartridge has an empty socket. The COMAL system can access an EPROM placed in that socket as OVERLAY 4 and OVERLAY 5.



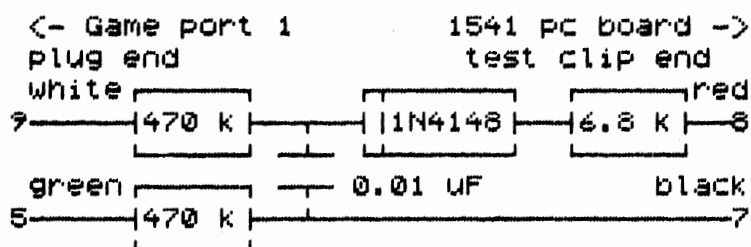
## 1541 ALIGNMENT

by Craig Van Degrift

This set of programs is for technically oriented people. They detail how to build low cost tools to use with your Commodore 64. With these cheap tools, you can use your Commodore 64 as replacement for an over thousand dollar oscilloscope. The colorful programs are interactive and very friendly. The programs also tell you how to work on a 1541 disk drive to do such things as adjust the alignment, set its speed, etc. Warning: you void your warranty if you open your disk drive. One technician we showed the program to said it was the only program he's seen that not only let's you work on a 1541 disk drive, but even helps you with the work. The picture below is a screen dump of one of the screens.

### HOW TO BUILD A VIDEO DETECTOR

1. Cut joystick extension cable 60 cm from end which plugs into the C64.
2. Cut the test clips and 20 cm of wire from the test leads.
3. Carefully solder together like this:



4. Cover exposed leads with heat shrink tubing in a way which avoids shorts
5. Be suspicious of the color coding in the joystick cable. You can check the port wiring diagram in Appendix I of the C64 Programmers Reference Guide.
6. See the Peltier books for how to make a similar circuit.

## CORRECTION TO DISK EDITOR

(on TODAY DISK #5)

The disk editor program on TODAY DISK #5 had a couple errors. First, EDITING: if the cursor is moved to line 11, and then scrolled down to the next line (12), the contents of the next line will not be picked up by the INPUT statement. To remedy this:

```
DELETE: 1340 INPUT " ... ": LINE$
ADD:    1340 PRINT " ... "
ADD:    1341 INPUT "": LINE$
```

The next 2 lines must also be modified:

```
1350 IF LINE$<>" " AND LINE$<>" " THEN
EDIT'LINE
1360 UNTIL LINE$="" OR LINE$=" "
```

Also, MSD Drives: All the COM\$ strings are allowed by the 1541 operating system, but I understand that the MSD drives are more particular about the punctuation. To enable use on the MSD drives merely punctuate these strings as follows:

```
COM$:="U2 2 0 "+TR$+" "+SC$
changes to:
COM$:="U2:2,0,"+TR$+", "+SC$
```

Disk Block: Track #18 Sector#1

Display and edit half block

```
00: 12 04 82 11 00 48 49 a0 - hi
08: a0 a0 a0 a0 a0 a0 a0 a0 -
10: a0 a0 a0 a0 a0 00 00 00 -
18: 00 00 00 00 00 00 14 00 -
20: 00 00 81 11 13 3e 2d 2d - >--
28: 2d 50 52 4f 47 52 41 4d --program
30: 53 2d 2d 2d 3c 00 00 00 -s---<
38: 00 00 00 00 00 00 01 00 -
40: 00 00 82 13 00 31 35 32 - 152
48: 30 44 45 4d 4f 31 a0 a0 -@demo1
50: a0 a0 a0 a0 a0 00 00 00 -
58: 00 00 00 00 00 00 07 00 -
60: 00 00 82 13 05 31 35 32 - 152
68: 30 44 45 4d 4f 32 a0 a0 -@demo2
70: a0 a0 a0 a0 a0 00 00 00 -
78: 00 00 00 00 00 00 07 00 -
```

## COMAL 2.0 DISK EDITOR

by Ian MacPhedran

This is an upgraded version of the disk editor previously presented in COMAL TODAY #5. This new version is on TODAY DISK #7. The major changes are:

- 1) Before the disk block is displayed, the user is asked if they wish the block to be printed. The default is YES, so type N and <RETURN> if this is not desired.
- 2) While editing, the user is asked for the line number. This is the address of the first character to be edited. Type in the HEX number and return. A line of eight HEX numbers will be displayed. Edit these and hit <RETURN>. Be sure to return to the main menu and re-display the block before writing to disk.
- 3) An added option is to read in the next block. This option is given in the menu.

## ELIZA FOR 2.0 AND 0.14

by Kevin Quiggle

"Eliza" is a somewhat simplified version of Joseph Weizenbaum's famous artificial intelligence program. There were originally different versions of this program; one of those versions, a psychotherapist "script" known as DOCTOR, is the basis for this particular COMAL adaptation.

Both the COMAL 2.0 and 0.14 versions are based on a simplified version of DOCTOR. A more complete version is published in "Artificial Intelligence for Small Computers" by John Krutch, together with many other interesting programs (in BASIC, unfortunately).

One final note: The COMAL 0.14 version of ELIZA makes use of the RESTORE LABEL procedure published in COMAL TODAY #6. Now how did Captain COMAL and his friends know I needed that procedure?

## CONTROL P FOR COMAL 0.14

by Steve Bruhn and Jesse Knight

COMAL 2.0 Cartridge users are lucky, they have a built in text screen dump that can be activated at any time by pressing CONTROL and P. Now, COMAL 0.14 users can have this same capability. Simply RUN the program listed below, answer the question with a reply of 15 (unless you have other machine code stored there), and the text dump routine is stored and protected. You then can SAVE, LOAD, EDIT and RUN other COMAL programs without disturbing it.

```
//text screen dump for 0.14
//original ml by Steve Bruhn
//COMAL 0.14 adaptation and relocater
//by Jesse Knight
//
repeat
  input "what page in $c000 (0-15) ": mlp
  age //wrap line
until mlpage>=0 or mlpage<16
mlpage:=mlpage+192
address:=mlpage*256
for i:=0 to 229 do
  read a
  if a=-1 then a:=mlpage
  poke address+i,a
endfor i
sys address
print "screen dump active"
print "use ctrl-p to dump the screen"
data 120,169,18,141,143,2,169,-1
data 141,144,2,169,0,141,232,-1
data 88,96,173,232,-1,240,3,76
data 72,235,173,141,2,201,4,240
data 3,76,72,235,165,203,201,41
data 240,3,76,72,235,238,232,-1
data 32,204,255,169,1,162,4,160
data 255,32,186,255,169,0,160,0
data 162,0,32,189,255,32,192,255
data 162,1,32,201,255,169,0,141
data 229,-1,141,230,-1,133,251,173
data 136,2,133,252,169,13,32,210
data 255,173,24,208,41,2,240,8
data 169,17,32,210,255,76,117,-1
data 169,145,32,210,255,160,0,140
data 231,-1,177,251,16,5,41,127
data 238,231,-1,201,32,176,5,105
data 64,76,159,-1,201,64,176,3
```

```
data 76,159,-1,201,96,176,5,105
data 128,76,159,-1,24,105,64,174
data 231,-1,240,12,72,169,18,32
data 210,255,104,32,210,255,169,146
data 32,210,255,230,251,208,2,230
data 252,238,229,-1,173,229,-1,201
data 40,208,178,169,13,32,210,255
data 169,0,141,229,-1,238,230,-1
data 173,230,-1,201,25,208,158,169
data 1,32,195,255,32,204,255,169
data 0,141,232,-1,96,0
```

## LOADING FONTS

by David Stidolph

COMAL 2.0 allows you to use custom designed character sets with your own programs. They can also be 'attached' to your program and be loaded and saved with them. However, standard COMAL 2.0 does not allow you to load a new character set from a running program.

The following procedure will allow you to load a standard COMAL character set without having to resort to using BATCH files. It takes approximately 11 seconds to load, and it presumes that you have ALREADY linked a character set to the program (otherwise an error will result).

```
PROC load'font(filename$) CLOSED
  IMPORT putcharacter
  TRAP
    OPEN FILE 79,filename$,READ
    FOR bank:=0 TO 1 DO
      FOR char:=0 TO 255 DO
        putcharacter(bank,char,GET$(79,8))
      ENDFOR char
    ENDFOR bank
  HANDLER
    CLOSE FILE 79
    REPORT // pass error outside
  ENDTRAP
  CLOSE FILE 79
ENDPROC load'font
```

## COMAL 2.0 FAST DIRECTORY READ

by George Jones

The following program will load the directory into memory faster than any other COMAL program, as well as being shorter than most. This is accomplished by using GET\$ to read in a large block of information, and the IN function to get at the specific information needed.

The program is also an example of doing things the easy way. Look at the old directory reading routines, and they look like converted BASIC programs [ed. note: they were], while this program retains the "flavor" of COMAL - readability.

```
// delete "read'directory"
// save "read'directory"
// copyright 1985 george jones
//
// Permission to reprint in
// COMAL TODAY is hereby granted,
// provided this notice is
// included in the reprinted
// material.
//
dim'stmnts
get'dir
//
PROC dim'stmnts
  DIM dir'entry$ OF 32
  DIM link$ OF 1
  DIM file'type$(144) OF 1
  DIM filesize$(144) OF 1
  DIM filename$(144) OF 16
ENDPROC dim'stmnts
//
PROC get'dir
  count#:=1
  sector#:=1
  MOUNT
  OPEN FILE 2,"u8:#2/s2/t+/d-"
  REPEAT
    setblock
    get'entries
  UNTIL sector#>18
  CLOSE FILE (2)
ENDPROC get'dir
//
```

```
PROC get'entries
  LOOP
    dir'entry$:=GET$(2,32)
    link$:=dir'entry$(2)
    check'link
    file'type$(count#):=dir'entry$(3)
    filename$(count#):=dir'entry$(6:21)
    PRINT filename$(count#)
    filesize$(count#):=dir'entry$(31)
    EXIT WHEN EOF(2)
    count#:+1
  ENDLOOP
ENDPROC get'entries
//
PROC setblock
  PASS "u1: 2 0 18 "+STR$(sector#)
  PASS "b-p: 2 0"
ENDPROC setblock
//
PROC check'link
  IF (ORD(link$)>1)=TRUE THEN
    sector#:=ORD(link$)
  ENDIF
ENDPROC check'link
```

## MAKING BATCH FILES

by David Stidolph

In COMAL Today #6 I talked about using batch files for setting up the function keys, but I did not tell how to create those files. The following program allows you to make a file containing up to 25 commands. The program will ask for a file name (I suggest you put "bat." in front to show it's purpose), and give some instructions on what to do. Enter each command on a separate line. When all the commands you want are on the screen in the order you want (top to bottom) press the STOP key, and the program will read the screen and create the batch file.

Batch files are one of the most powerful features of 2.0 COMAL. They give the ability for a program to modify itself, and to perform a lengthy task (like LISTing an entire disk of programs to a printer) by itself.

More

```

// delete "0:batchfile'maker"
// save "0:batchfile'maker"
//
USE system
DIM filename$ OF 12
DIM a$(1:25) OF 40
//
PAGE
PRINT "Create a BATCH execute file ..."
PRINT
PRINT "You can create an auto exec file
of" //wrap line
PRINT "up to 25 commands easily."
PRINT
PRINT "Simply type on the screen the com
mands" //wrap line
PRINT "that you would like in the exec f
ile." //wrap line
PRINT "when done --- just hit the ",
textcolors(-1,-1,7)
PRINT "STOP ",
textcolors(-1,-1,1)
PRINT "key."
PRINT
PRINT "This program will read what is on
the" //wrap line
PRINT "screen and create a batch file fo
r you." //wrap line
PRINT AT 15,1: "Enter up to 12 letter fi
le name > " //wrap line
textcolors(-1,-1,7)
INPUT AT 17,5,12: "bat.": filename$
textcolors(-1,-1,1)
PRINT AT 19,1: "To use: SELECT INPUT ""b
at."+filename$+""" //wrap line
INPUT AT 22,1: "Hit return to start maki
ng bat."+filename$+": ": dummy$ //wrap
line
//
PAGE
TRAP ESC-
REPEAT
    PRINT inkey$,
UNTIL ESC
//
OPEN FILE 1,"ds:",READ
//
FOR x:=1 TO 25 DO
    CURSOR x,1
    INPUT FILE 1: a$(x)
ENDFOR x
//
CLOSE FILE 1

```

```

TRAP
    SELECT OUTPUT "0:bat."+filename$
    FOR x:=1 TO 25 DO
        IF a$(x)<>SPC$(39) THEN PRINT a$(x)
    ENDFOR x
HANDLER
    SELECT OUTPUT "ds:"
    PRINT ERRTXT$
    END "File aborted!"
ENDTRAP
SELECT OUTPUT "ds:"
//
TRAP ESC+
END "Batch File: bat."+filename$+" creat
ed!" //wrap line ■

```

## DISASSEMBLER IN COMAL 2.0

by Ian MacPhedran

This program enables the user to examine memory locations in the Commodore 64, the COMAL 2.0 Cartridge, the 1541 disk drive, as well as programs on diskette. This is NOT an editor, merely an advanced version of a memory peeker.

The first prompt the user will receive on RUNing the program is whether the printer is to be used to output the disassembly. Answer Y or N.

The computer then pauses while the opcode mnemonics are loaded, and then comes up with a menu screen. The six options are:

a) Load memory from computer. This loads a block of memory from the computer or cartridge. The prompt for ROM page gives locations as follows: 0 = hidden RAM; 1 = Page 1 of cartridge; 2 = Page 2; 3 = Page 3; 4 = Page 4. Any other number will give no change.

B) Load memory from disk drive. This will load a block of memory from the drive's RAM or ROM. Do not give a starting address of greater than \$FF00 (hex) or 65280 (decimal).

More ■

C) Load a disk block. This loads a block from the disk. Note for disassembly: the disassembler ignores the link vector (first two locations on block) and takes the second pair of bytes of the first block as the starting address.

D) Disassemble. This disassembles the loaded block of memory into standard MOS mnemonics. To disassemble the next line, press a key. To proceed continuously through the block, press a repeating key such as a cursor key, or space bar. This disassembly will continue over block boundaries by automatically reading in the next block, so be careful when examining the last block of a disk file or the \$FF00 block of memory (computer or drive). To exit the disassembler, press the E, X, or Q keys.

E) Display half block. This gives a HEX/ASCII dump of the block contents. As only half a block will fit on the screen at one time, you must choose which half you want.

F) Continue. This reads in the next block. This allows you to skip ahead, for example, through blocks of text, until you decide to disassemble. Remember that a disk block will be missing the first four bytes when being disassembled, and that a strange address may be given if disassembly starts on a block other than the first.

G) Quit. Exit the program.

Disassembly Computer memory \$9000 +  
(Press 'q' to quit)

```

9000 cd 0c c2 cmp $c20c
9003 b0 04 bcs $9009
9005 cd 14 c2 cmp $c214
9008 60 rts
9009 18 clc
900a 60 rts
900b 20 89 95 jsr $9589
900e a0 01 ldy #$01
9010 b1 45 lda ($45),y
9012 49 80 eor #$80
9014 91 45 sta ($45),y
9016 60 rts

```

## MAKE DATA STATEMENTS FROM FILES

by T Creasey

This program will convert a PRG type object file of machine code into a COMAL program file consisting of DATA statements for the same code. This is useful if you have a machine code program on disk that you'd like to include in a COMAL program. Simply merge the DATA statements into your program, and add a short routine to READ them and POKE the values into the proper locations.

```

// delete "make'data'stmts"
// save  "make'data'stmts"
// by creasey
//

// 01 Mar 85
//
line:=10
checksum:=0
DIM data'line$ OF 80
DIM new'line$ OF 80
//
input'files
open'files
create'line("DATA ")
addr
read'ml'value
close'output
END "Finished."
//
PROC input'files
  INPUT "File to be converted? ": readfi
  le$ //wrap line
  INPUT "Data File to be created? ": dat
  afile$ //wrap line
ENDPROC input'files
//
PROC open'files
  CLOSE
  OPEN FILE 1,"0:"+readfile$,READ
  OPEN FILE 2,"0:"+datafile$,WRITE
ENDPROC open'files
//
PROC read'files
  temp$:=GET$(1,1)
  ascii:=ORD(temp$)
ENDPROC read'files

```

More



```

PROC create'line(command$)
  line'number$:="00"+STR$(line)
  l:=LEN(line'number$)
  new'line$:=line'number$(1-3:1)
  new'line$:"+ " +command$
  line:=line+10
ENDPROC create'line
//
PROC addr
  read'files
  lo'addr:=ascii
  read'files
  hi'addr:=ascii
  start'addr$:=STR$(lo'addr+256*hi'addr)
  new'line$:=new'line$+start'addr$
  new'line$:"+ " // start addr"
  store'line
  PRINT new'line$
ENDPROC addr
//
PROC read'ml'value
  WHILE NOT EOF(1) DO
    create'line("DATA ")
    data'line$:=""
    FOR counter:=1 TO 6 DO
      IF NOT EOF(1) THEN
        read'files
        ml'value$:=STR$(ascii)
        checksum:=checksum+ascii
        data'line$:=data'line$+ml'value$
        data'line$:=data'line$+","
      ENDIF
    ENDFOR counter
    l:=LEN(data'line$)
    new'line$:=new'line$
    new'line$:=data'line$(1:l-1)
    store'line
  ENDWHILE
  CLOSE FILE 1
ENDPROC read'ml'value

PROC store'line
  PRINT FILE 2: new'line$
ENDPROC store'line
//
PROC close'output
  create'line("DATA ")
  data'line$:=STR$(checksum)
  new'line$:=new'line$+data'line$
  new'line$:"+ " // checksum value"
  PRINT FILE 2: new'line$
  create'line("END")
  PRINT FILE 2: new'line$
  CLOSE FILE 2
ENDPROC close'output ■

```

## MAKE COMAL OBJECT FILE

by Ian MacPhedran

This program is useful to convert output from a non Commodore Assembler into the standard Commodore Assembler file output as required for use with the COMAL 2.0 LINK statement.

```

// delete "0:make'object'file"
// save  "0:make'object'file"
// Ian MacPhedran 1985
//
PAGE
DIM a$ OF 1, filein$ OF 16
DIM objfile$ OF 16, hx$ OF 16
DIM objstr$ OF 60
hx$:="0123456789abcdef"
PRINT "Input name of file to be converted" //wrap line
INPUT filein$
PRINT ""13"Thank you"13""
PRINT "Input name of output object file"
INPUT objfile$
PRINT ""13"Thank you"13""
TRAP
  OPEN FILE 2,filein$+",p,r/d+"
  OPEN FILE 3,objfile$+",s,w/d+"
HANDLER
  PRINT "ERROR: ",ERRTEXT$
  END
ENDTRAP
rec'num:=0
addr:=ORD(GET$(2,1))+256*ORD(GET$(2,1))
LOOP
  rec'num:+1
  i:=0
  checksum:=0
  adhi#:=addr DIV 256
  adlo#:=addr MOD 256
  checksum:=adhi#+adlo#
  objstr$:=hx$(adhi# DIV 16+1)+hx$(adhi# MOD 16+1) //wrap line
  objstr$:=objstr$+hx$(adlo# DIV 16+1)+hx$(adlo# MOD 16+1) //wrap line
  REPEAT
    i:+1
    byt#:=ORD(GET$(2,1))
    checksum:=byt#
    IF EOF(2) THEN EXIT
    objstr$:=objstr$+hx$(byt# DIV 16+1)+hx$(byt# MOD 16+1) //wrap line
  UNTIL i=$18

```

More ■

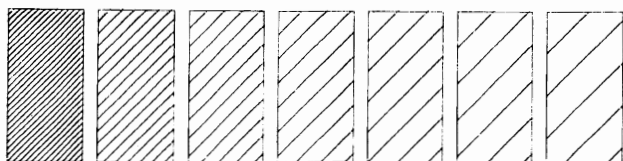
```

checksum:=$18
checksum:=checksum MOD 216
ckhi#:=checksum DIV 256
cklo#:=checksum MOD 256
objstr$:=hx$(ckhi# DIV 16+1)+hx$(ckhi#
MOD 16+1) //wrap line
objstr$:=hx$(cklo# DIV 16+1)+hx$(cklo#
MOD 16+1) //wrap line
addr:=$18
PRINT FILE 3: ";18"+objstr$
ENDLOOP
i:-1
IF i<>0 THEN
checksum:=i
checksum:=checksum MOD 216
ckhi#:=checksum DIV 256
cklo#:=checksum MOD 256
objstr$:=hx$(ckhi# DIV 16+1)+hx$(ckhi#
MOD 16+1) //wrap line
objstr$:=hx$(cklo# DIV 16+1)+hx$(cklo#
MOD 16+1) //wrap line
objstr$:= ";" +hx$(i DIV 16+1)+hx$(i MOD
16+1)+objstr$ //wrap line
PRINT FILE 3: objstr$
ENDIF
objstr$:= ";00"
rechi#:=rec'num DIV 256
reclo#:=rec'num MOD 256
objstr$:=hx$(rechi# DIV 16+1)+hx$(rechi#
MOD 16+1) //wrap line
objstr$:=hx$(reclo# DIV 16+1)+hx$(reclo#
MOD 16+1) //wrap line
objstr$:=hx$(rechi# DIV 16+1)+hx$(rechi#
MOD 16+1) //wrap line
objstr$:=hx$(reclo# DIV 16+1)+hx$(reclo#
MOD 16+1) //wrap line
PRINT FILE 3: objstr$
CLOSE FILE 3
CLOSE FILE 2
END ■

```

HATCHING

Program: 1520demo10



## 1520 PLOTTER DRIVER PROCEDURES

by Kevin Quiggle

If you have a Commodore 1520 plotter, these procedures will allow you to take advantage of it many features from COMAL. Just merge the ones you need into your programs. These procedures as well as many demonstration programs are on TODAY DISK #7 for both 0.14 and 2.0 versions of COMAL.

This is the 0.14 Plotter Driver:

```

//save "1520driver0.14
// by kevin quiggle
//
proc p'open(sa) closed
open file 66,"",unit 6,sa,write
endproc p'open
//
proc p'close closed
close file 66
endproc p'close
//
proc p'char(c$)
p'open(6)
print file 66: 1,
p'close
p'open(0)
if c$<>"" then
print file 66: c$,
else
print file 66:
endif
p'close
endproc p'char
//
proc p'home
p'open(1)
print file 66: "h",
p'close
endproc p'home
//
proc p'init
p'open(1)
print file 66: "i",
p'close
endproc p'init
//

```

More ■

```

proc p'moveto(x,y)
  p'open(1)
  command("m",x,y)
  print file 66: c$,
  p'close
endproc p'moveto
//
proc p'drawto(x,y)
  p'open(1)
  command("d",x,y)
  print file 66: c$,
  p'close
endproc p'drawto
//
proc p'move(x,y)
  p'open(1)
  command("r",x,y)
  print file 66: c$,
  p'close
endproc p'move
//
proc p'draw(x,y)
  p'open(1)
  command("j",x,y)
  print file 66: c$,
  p'close
endproc p'draw
//
proc p'reset
  p'open(7)
  print file 66:
  p'close
endproc p'reset
//
proc p'color(color)
  p'open(2)
  convert(2,t$,color)
  print file 66: t$,
  p'close
endproc p'color
//
proc p'charsize(siz)
  p'open(3)
  convert(2,t$,siz)
  print file 66: t$,
  p'close
endproc p'charsize
//
proc p'rotchar(rot)
  p'open(4)
  convert(2,t$,rot)
  print file 66: t$,
  p'close
endproc p'rotchar

```

```

//
proc p'scribe(brk)
  p'open(5)
  convert(2,t$,brk)
  print file 66: t$,
  p'close
endproc p'scribe
//
proc convert(an,ref alph$,ref numb) clos
ed //wrap line
  z:=zone
  zone 0
  dim old$ of 12
  row:=peek(214); col:=peek(211)
  pc:=peek(646) //current pencolor
  bc:=peek(53281)-240
  pencolor bc
  print chr$(19),
  open file 126,"",unit 3,read
  input file 126: old$
  close file 126
  print "          ",
  case an of
  when 1
    if alph$="" then
      alph$="nv"
    else
      print alph$,
    endif
  when 2
    print numb,
  when 3
    if numb>=0 and numb<=255 then
      print chr$(numb),
    else
      alph$="nv"
    endif
  otherwise
    alph$="nv"
  endcase
  if alph$<>"nv" then
    pull'screen(an,alph$,numb)
  endif
  pencolor pc
  print "",old$
  poke 214,row
  poke 209,(1024+row*40) mod 256
  poke 210,(1024+row*40) div 256
  poke 211,col
  zone z
endproc convert
//
More

```

```

proc pull'screen(an,ref alpha$,ref numbe
r) closed //wrap line
print chr$(19),
open file 93,"",unit 3,read
case an of
when 1
input file 93: number
when 2,3
input file 93: alpha$
endcase
close file 93
endproc pull'screen
//
proc command(ch$,x,y)
c$:=ch$+" "
convert(2,t$,x)
if " " in t$ then t$:=t$(1:" " in t$)
c$:=c$+t$+" "
convert(2,t$,y)
if t$(1)<>"0" then
c$:=c$+t$
else
c$:=c$+t$(2:len(t$))
endif
endproc command

```

This is the 2.0 Plotter Driver:

```

// delete "proc.1520'driver"
// list "proc.1520'driver"
// by kevin quiggle
//
PROC p'open(sa) CLOSED
SELECT OUTPUT "u6:/s"+STR$(sa)
ENDPROC p'open
//
PROC p'close CLOSED
SELECT OUTPUT "ds:"
ENDPROC p'close
//
PROC p'char(c$)
p'open(6)
PRINT 1
p'open(0)
IF c$<>" " THEN
PRINT c$,
ELSE
PRINT c$
ENDIF
p'close
ENDPROC p'char
//

```

```

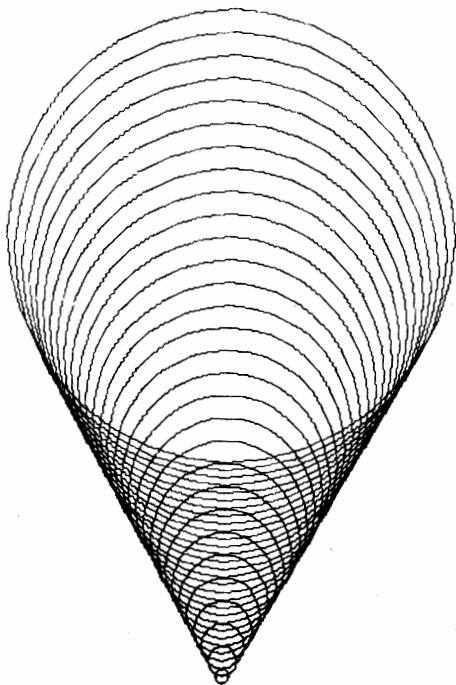
PROC p'home
p'open(1)
PRINT "h"
p'close
ENDPROC p'home
//
PROC p'init
p'open(1)
PRINT "i"
p'close
ENDPROC p'init
//
PROC p'moveto(x,y)
p'open(1)
PRINT "m";x;CHR$(29);y;CHR$(29)
p'close
ENDPROC p'moveto
//
PROC p'drawto(x,y)
p'open(1)
PRINT "d";x;CHR$(29);y;CHR$(29)
p'close
ENDPROC p'drawto
//
PROC p'move(x,y)
p'open(1)
PRINT "r";x;CHR$(29);y;CHR$(29)
p'close
ENDPROC p'move
//
PROC p'draw(x,y)
p'open(1)
PRINT "j";x;CHR$(29);y;CHR$(29)
p'close
ENDPROC p'draw
//
PROC p'reset
p'open(7)
PRINT
p'close
ENDPROC p'reset
//
PROC p'color(color)
p'open(2)
PRINT color
p'close
ENDPROC p'color
//
PROC p'charsize(size)
p'open(3)
PRINT size
p'close
ENDPROC p'charsize

```

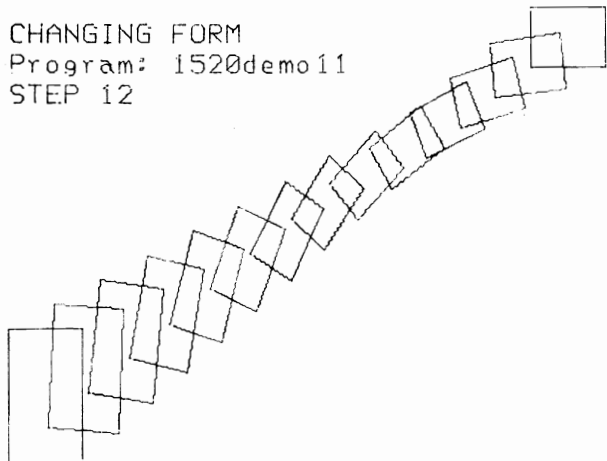
More 

```
//
PROC p'rotchar(rot)
  p'open(4)
  PRINT rot
  p'close
ENDPROC p'rotchar
//
PROC p'scribe(brk)
  p'open(5)
  PRINT brk
  p'close
ENDPROC p'scribe ■
```

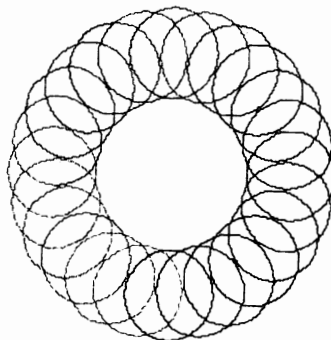
CONE MADE FROM CIRCLES  
Program: 1520demo4



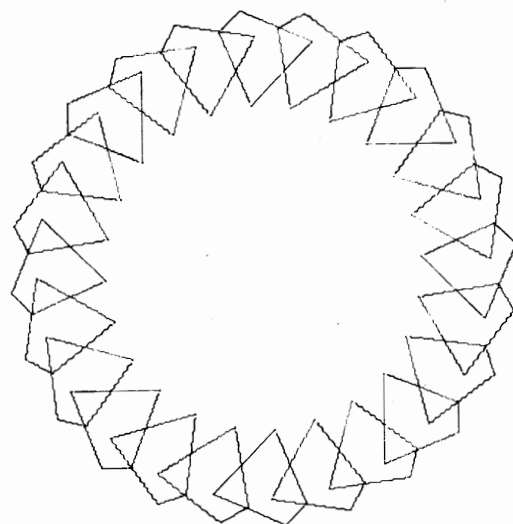
CHANGING FORM  
Program: 1520demo11  
STEP 12



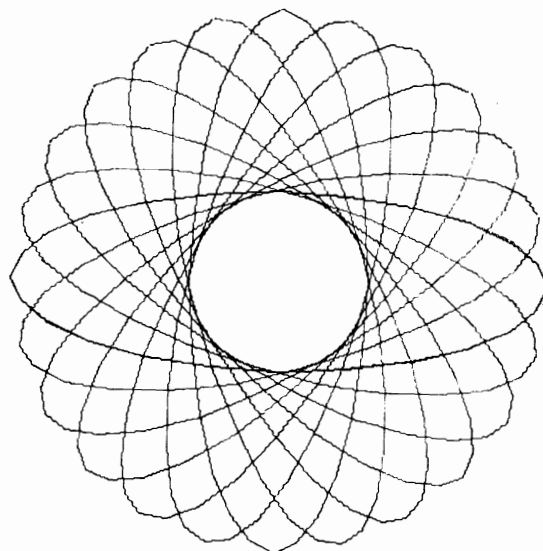
CIRCLE MADE FROM CIRCLES  
Program: 1520demo5



ROTATING FIGURES  
Program: 1520demo12



ROTATING ELLIPSE  
Program: 1520demo6



## GEMINI 10X GRAPHICS DUMP

This program puts machine code into the RS-232 buffers. The machine code can dump a HI-RES graphics screen to the Gemini 10X printer using the Cardco/A? interface (dip switch #2-2 & 4 = OFF for no linefeed). The printing starts at the leftmost margin. You can call this routine in four ways:

SYS 49152 = normal size  
 SYS 49155 = normal size reversed  
 SYS 49160 = double width  
 SYS 49163 = double width reversed

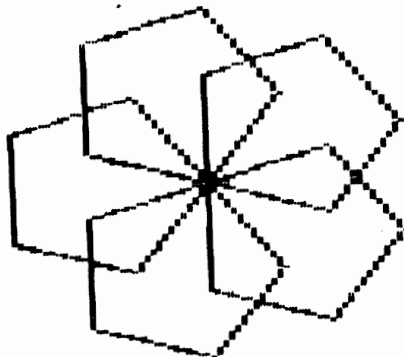
```
// delete "gem10x'card/a"
// save  "gem10x'card/a"
//
// graphics dump using card/?A
// & gemini 10x
//
// 02 mar 85
```

```
ml'value:=0
checksum:=0
//
READ address
LOOP
  READ ml'value
  EXIT WHEN ml'value>256
  POKE address,ml'value
  checksum:=checksum+ml'value
  address:=address+1
ENDLOOP
IF checksum=ml'value THEN
  mes$:="SYS 49152 to dump graphics."
ELSE
  mes$:="Checksum error!!"
ENDIF
END mes$
//
DATA 49152 // start address
DATA 76,31,192,169,255,76,33
DATA 192,76,15
DATA 192,169,255,208,2,169,0
DATA 141,126,192
DATA 169,121,160,192,162,3
DATA 142,131,192,208
DATA 11,169,0,141,126,192,169
DATA 75,160,64
DATA 162,1,141,249,192,140
```

```
DATA 250,192,142,251
DATA 192,165,250,141,254,192
DATA 32,16,193,169
DATA 40,133,251,162,0,189,248
DATA 192,240,6
DATA 32,210,255,232,208,245
DATA 32,191,192,169
DATA 0,162,7,157,8,193,202
DATA 16,250,162
DATA 0,160,0,189,0,193,57
DATA 233,192,240
DATA 9,185,8,193,29,233
DATA 192,153,8,193
DATA 200,192,8,208,234,232
DATA 224,8,208,227
DATA 162,0,189,8,193,73,0
DATA 32,210,255
DATA 160,0,240,6,32
DATA 210,255,32,210,255
DATA 232,224,8,208,233
DATA 198,251,208,183,169
DATA 13,32,210,255,32,225
DATA 255,240,4,198
DATA 250,208,152,169,27
DATA 32,210,255,169,64
DATA 32,210,255,173,254
DATA 192,133,250,173,255
DATA 192,133,251,169,4,32
DATA 195,255,32,204
DATA 255,120,169,52
DATA 133,1,162,0,169,0
DATA 157,0,193,173,0,224
DATA 157,0,193,238
DATA 204,192,232,224,8
DATA 208,237,173,204,192
DATA 201,0,208,3,238,205
DATA 192,169,55,133
DATA 1,88,96,128,64,32,16,8,4,2
DATA 1,27,64,13,27,51,16,0,27,75
DATA 64,1,0,0,0,0,0,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0
DATA 0,0,165,251,141,255
DATA 192,169,0,141
DATA 204,192,169,224,141
DATA 205,192,169,25,133
DATA 250,169,4,162,4,160
DATA 4,32,186,255
DATA 169,0,32,189,255,32
DATA 192,255,162,4
DATA 32,201,255,162,0,189
DATA 241,192,240,6
DATA 32,210,255,232,208,245,96
DATA 42661 // checksum value
```

## POLYGONS - A PROGRAM

```
// delete "polygons"
// save  "polygons"
// by Bob Hoerter
//
USE turtle
init
FOR t:=angle TO 360 STEP angle DO
  moveto(0,0)
  setheading(t)
  box(number'of'sides,length'of'side)
ENDFOR t
//
WHILE KEY$<>"" DO NULL
WHILE KEY$="" DO NULL
END "Did you like that ?"
//
PROC box(no'sides,length'of'side)
  FOR side:=1 TO no'sides DO
    fd(length'of'side)
    rt(angle)
  ENDFOR side
  hideturtle
ENDPROC box
//
PROC get'data
  PAGE
  INPUT "number of sides > ": number'of'
  sides //wrap line
  angle:=360 DIV number'of'sides
  INPUT "length of the side (50 to 100)
  >": length'of'side //wrap line
ENDPROC get'data
//
PROC init
  hideturtle
  nowrap
  get'data
  graphicscreen(1)
  viewport(50,250,10,190)
  window(-250,250,-250,250)
  pencolor(0)
  paint(0,0)
  pencolor(1)
ENDPROC init
```



## 2.0 QUICK TRICK FOR DATA READING

by Susan Long

Here is a trick that can save time in reading data statements. [Note: This only works for version 2.0] Suppose you have a program, such as an adventure game that contains a lot of data, perhaps of different types. Instead of reading these data statements into variable arrays at the beginning of the program, you can save memory by leaving them as data and reading them repeatedly during the program, as illustrated in the following example.

Assign a label to each section of data. Then when you need to read your values, use the RESTORE command, along with the label, to reset the data pointer to the beginning of the appropriate section. For example:

```
//
//
//
restore numberdata
for count = 1 to choice
  read number
  print choice
endfor count
//
restore textdata
for count = 1 to special
  read text$
  print text$
endfor count
//
//
//
numberdata:
data 0,2,3,0,5,6,0
data 1,0,3,0,0,13
//
//
//
textdata:
data "treasure vault"
data "coal mine"
//
//
//
```



## FIND RADICALS

by Steve Smullen

No, this is not a political program.  
Rather it is a math program to simplify  
complex radicals. It shows off the text  
graphics mode.

```
// delete "0:find'radical"
// save  "0:find'radical"
// By: Steve Smullen
//
USE graphics
//
REPEAT
    setup'screen'variables
    find'print'radical
UNTIL a$="x"
//
PAGE
PRINT CHR$(14) // lower case
END "READY."
//
PROC setup'screen'variables
    PAGE
    PRINT CHR$(142) // uppercase
    textbackground(0)
    textborder(0)
    PRINT AT 2,12: "reducing radicals"
    row:=11
    column:=5
    print'radical
    PRINT AT 9,1: "input indice"
    REPEAT
        INPUT AT 12,3,2: "": indice
    UNTIL indice>1
    PRINT AT 9,1: "input radicand"
    INPUT AT 12,6,6: "": rad
    PRINT AT 9,1: "      -      "
    numb:=1
    number:=1
    radical:=ABS(rad)
ENDPROC setup'screen'variables
//
PROC find'print'radical
    REPEAT
        numb:+1
    UNTIL radical<numb**indice
    numb:-1
```

```
REPEAT
    IF radical DIV numb**indice=radical/
    numb**indice THEN //wrap line
        number:=number*numb
        radical:=radical/numb**indice
    ENDIF
    numb:-1
UNTIL numb=0
PRINT AT 12,17: "="
IF radical>1 THEN
    CURSOR 12,24
    IF number>1 THEN
        PRINT number,
    ENDIF
    IF rad<1 THEN
        PRINT "i"
    ELSE
        PRINT
    ENDIF
    row:=11
    column:=30
    print'radical
    PRINT AT 12,28: indice
    PRINT AT 12,31: radical
ELSE
    PRINT AT 12,31: number,
    IF rad<1 THEN
        PRINT "i"
    ENDIF
ENDIF
PRINT AT 23,5: "press c to continue or
x to end" //wrap line
REPEAT
    a$:=KEY$
UNTIL a$="c" OR a$="x"
ENDPROC find'print'radical
//
PROC print'radical
    CURSOR row,column
    PRINT CHR$(111),
    FOR x:=1 TO 6 DO
        PRINT CHR$(183),
    ENDFOR x
    CURSOR row+1,column
    PRINT CHR$(165)
    CURSOR row+2,column-2
    PRINT CHR$(163),
    PRINT CHR$(109),
    PRINT CHR$(165)
ENDPROC print'radical
```

# Mailing List Maker

by Doug Drake

Ever since I started doing a User Group newsletter, I've been dreaming of the perfect Mailing List program. It's not that the MACUG mailing list is any great challenge to maintain, but more a matter of convenience. Many months ago I came to the conclusion that I wouldn't be happy unless I wrote my own program to do the job. Now, one disk drive later, it's done.

Mailing List is written in COMAL 0.14, but you don't have to know much about COMAL to use it. To get it up and running just LOAD "mailing'list" and RUN. You'll have to wait about 30 seconds while the program loads before you get the first menu.

There are two menus. The first asks whether you want to create a new file or use an existing file. Relative files are used. For a new file you give it a name, and estimate the number of records. For an existing file you just give the name. You can also exit the program from this menu, but you'll still be in COMAL. Then to go to Basic just issue the command: BASIC.

After a new file is created, or you give the name of an existing file, you get a second menu. This menu allows you to enter a new address, delete an address, edit an address, preview the addresses on the screen, and print addresses -- either on mailing labels or paper.

The following fields are allowed: first and last names, organization name, street address, city, state, zip, and other information. There is also a print flag, so you can have a name in the file but not print it. For example, my name is a part of the MACUG file but when I mail

the newsletter I don't need a mailing label for myself. The organization name isn't printed unless there's something there. I use the "other" field for phone numbers, and for a code that says whether someone paid their dues.

Also included are three kinds of sorts, for re-arranging your files. All are based on the Quicksort routine. Some of these sorts are pretty slow because writing relative files is slow, but if you watch the screen you'll see that the Quicksort part is very fast. Each address is read from the disk as it's needed, rather than having all addresses in RAM in an array. This means it's possible to have some pretty big lists with this program if need be. But it does slow down some of the sorting.

When you want to sort, a second program module is loaded. You can sort an existing file and re-write it on the disk; you can sort a file and write a second file with a new name to disk; or you can create an index file which will be referenced when you preview addresses on the screen or print addresses. The latter is a fast sort because the records on disk are not rearranged. Printing is a little slower though. You can sort by name, organization, zip, or other.

I've done my best to make the program self-explanatory, without going overboard. There is only one bug that I know of, and you're not likely to run into it. If you create a file with only one anticipated record, the last name won't be written to the record. This only happens in the one particular case. It would still be a good idea to check the file with the preview option after you add addresses.

More 

The edit function could use a little additional explanation. You're asked for the target name you want to edit. If you just hit Return you'll step through every record. Just type "N" to go on to the next record. That way you can find a record even if you forget the last name. The program can print mailing labels either in all capital letters or in caps/lower case. When you enter the address, you won't be allowed to enter any capital letters. The small letters you enter would be capitals on the printer. If you want caps/lower case, you'll have to go to the edit function, when you will be allowed to enter capitals. I know this is inconvenient, but allowing caps to be entered in the new address routine (called Take'in in the listing) really slowed things down. So it's a compromise.

There are a couple of interesting procedures (subroutines) I developed for this program. One is used for editing and is called Screen'prompt. The existing value for each of the fields is printed on the screen. You can use the editing keys to make changes (but stay in the brackets) and then hit Return. If you don't want to make a change, just hit Return.

The other interesting routine is the final (I think) solution to a relative file bug that popped up in COMAL because it's faster [than BASIC]. I've mentioned something about this before, but in a nutshell every once in awhile a field doesn't get written because the disk drive's buffer doesn't empty before more data comes down the wire from the computer. The drive doesn't tell the computer to wait. This happens when a record crosses a block boundary. The Write'record routine in this program calculates whether each field will cross such a boundary before it is actually written. If it will encounter the bug, another procedure called Disk'pause is called to let the computer catch up. While the routine is specific to this program, it illustrates the solution and could be applied to other programs.

There are a few unfriendly things which can happen in the program and I want to warn you about them. When you delete an address, it's possible for some strange things to happen. Even though it's been deleted and won't print, it is still there until you re-sort the file with either Sort #1 or Sort #2. An index file will continue to work, until you do one of the other sorts -- then it will go bad. On the other hand, if you add an address without re-doing your index file, you'll have a problem too.

When the sorts get messed up this way, you might get dumped into COMAL with an error message. In my new cartridge COMAL 2.0 I can trap these errors, but not in the disk version. Here's what to watch out for:

If you re-sort using Sort #1 or Sort #2, but do not re-do an Index File (Sort #3), you'll get Error 11. If the red light on the drive is on, type Close and hit Return. Then Run and do the Index Sort to restore things to normal.

If you've re-done the Index File without doing Sort #1 or #2 (after you deleted an address), you'll get a message from within the program. You'll have to do Sort #1 or #2, then re-do the Index Sort.

If you add an address without re-doing the Index, you'll get a message from within the program, as above.

And if you enter a filename that doesn't exist, you'll get Error 8. Type Close and Return. Then type Cat and Return to see the disk directory and find the right name. Then type Run and you're back in the program.

This probably isn't the perfect program for everyone, but if you need a mailing list I'd suggest you give it a try. ■

# Pressure Drops and Water Flow

by Lowell Zabel

We wanted to install a sprinkler system in our lawn so that we could have green grass during the dry season. Being frugal types, we didn't want to install larger pipes than necessary to carry the sprinkler head's capacity nor did we want to use such small pipe that there was little pressure left when the water reached the head.

The obvious approach was to calculate the pressure drops. This can't be done directly so the problem was an ideal one for computer use because of the iterative abilities of the computer. We were pumping water, but since viscosity and specific gravity terms of the pumped liquid are explicit in the equations, these were included as input elements. It becomes easy for water since both the viscosity and specific gravity are one at normal temperatures.

Data entry is through input statements. The data required are the pipe inside diameter, the flow, the specific gravity of the liquid and its viscosity. The pressure drop also depends on the roughness of the inside of the pipe. Pipe type is selected from a short menu and the computer obtains the roughness from a look-up table.

Liquid flow exists in three main regimes; laminar, transition and turbulent. This program is valid only for turbulent flow, where the Reynold's number is greater than 4000. The calculations are straight forward except for the Fanning friction factor. The friction factor cannot be separated from the rest of the equation. Therefore, an estimate is made and the difference between two calculated values of F is computed. If the difference is not acceptably small, a new estimate is

made and a new difference calculated. Once the error is low enough, it proceeds to the pressure drop calculation. The display shows the pipe size, pipe material and the pressure drop for 100 feet of pipe. In order to obtain the system pressure drop, the effects of fittings such as elbows and tee's must be included. These effects are not a part of this program.

Example: 100 feet of 2 inch ID drawn tubing and a flow of 25 GPM of water results in a pressure drop of 11 PSI.

```
// delete "flow"
// save  "flow"
//  by Lowell Zabel
//
// This program calculates the
// pressure drop in a pipe
// carrying fluid.
//
variables
pipe'type
calculations
END "End of program"
//
PROC variables
PAGE
INPUT "Pipe ID in inches = ": d
INPUT "Flow in GPM = ": q
INPUT "Specific Gravity of fluid=": sg
INPUT "Viscosity of fluid in CP = ": u
PRINT
ENDPROC variables
//
PROC pipe'type
PRINT "1 Drawn Tubing"
PRINT "2 Steel of Iron Pipe"
PRINT "3 Galvanized Iron Pipe"
PRINT "4 Cast Iron Pipe"
PRINT "5 Concrete Pipe"
PRINT
PRINT "Select type - enter number";
INPUT "then press RETURN:": m
```

More ➡

```

CASE m OF
WHEN 1
  e:=5e-06
  type$:="drawn tubing"
WHEN 2
  e:=1.5e-04
  type$:="steel or iron pipe"
WHEN 3
  e:=5e-04
  type$:="galvanized pipe"
WHEN 4
  e:=8.5e-04
  type$:="cast iron pipe"
WHEN 4
  e:=5e-03
  type$:="concrete pipe"
OTHERWISE
  pipe'type
ENDCASE
ENDPROC pipe'type
//
PROC calculations
  PRINT ""147""
  //calculate e/d
  ed:=e/d*12
  //calculate velocity
  v:=(1/d)*.4085*q
  //calculate Reynolds number
  n:=v*d*sg*64.4/v/8.064e-03
  //calculate Fanning factor
  f:=.125/(n32)
  fanning
PROC fanning
  x:=1/SQR(f)+2*LOG((1/SQR(f))*2.51/n)+
  ed/3.7)/LOG(10) // wrap line
  IF ABS(x)>=1e-04 THEN
    f:=f*(1+.1*x)
    fanning
  ENDIF
ENDPROC fanning
//
//calculate head loss
h:=18.633*v2 / d
p:=14.7*h/32.2
PRINT "For 100 ft. of ";d;"in.";type$
PRINT
PRINT "with a flow of ";q;" GPM"
PRINT
pressure$:="the head loss =###.## ft."
PRINT USING pressure$: h
PRINT
pressure$:="or ###.## psi."
PRINT USING pressure$: p
ENDPROC calculations

```

## Foreign Orders

We are set up to handle orders from the United States and Canada only. Orders from other countries are discouraged, but accepted if extra shipping and handling is included, and payment is in US Dollars drawn on a US bank. Foreign orders add the following extra charges:

- \$3 per disk
- \$5 per Captain COMAL book
- \$8 per Captain COMAL book / disk set
- \$10 per other book
- \$30 per 6 issue newsletter subscription

## GERMAN COMAL GROUP

Now there is one central German COMAL Users Group, complete with a newsletter and over 2000 members:

Christiane and Gerhard Canisius  
 D-400 Dusseldorf 12  
 Freiheitstrasse 30  
 West Germany  
 phone: 0211-27 93 36

## CHECK CARTRIDGE PROGRAM FIX

The program CHECK'CARTRIDGE on Cartridge Demo Disk #3 was written specifically for the gray EPROM cartridge. If you have the black ROM cartridge the program will not function properly. It will tell you that 3 of 4 chips are bad, when you really only have two chips, and they both are probably good. We updated the program and listed the new version in COMAL TODAY issue #6 on page 40. The program was included on the TODAY DISK #6. We are also including it on TODAY DISK #7 in case new subscribers need it. The program will test every bit in your cartridge and tell you if it is 100% good.

# Differential Equations

by Lowell Zabel

Engineers and scientists have a propensity for describing physical systems and phenomena in mathematical terms. Hence there is a need to solve these equations. Static systems are adequately described by algebraic equations. The solutions to these equations are not covered here. Whenever variations or movement is inherent in the physical system, the resulting descriptive equations are differential equations. Differential equations can be divided into two classes, ie, ordinary and partial. These can be further divided into linear and non-linear. For the present we will look at ordinary differential equations. These have the form  $ax'''+bx''+cx'+dx=f$  where the coefficients, a, b, c & d may be constants or functions of x and or t the independent variable. F is the forcing function which may also be a function of x and or t. The order of the equation is determined by the highest derivative. The example is a third order equation.

When all coefficients are constants, the equations are readily solved using Laplace transforms although the work for the solutions of higher order equations can be quite involved. Non-linear equations usually are very difficult to solve and many times are not attempted. Before continuing, I should point out that f usually takes one of four forms. The first is the Dirac impulse. This is a pulse having infinite amplitude, zero length and unit area under the curve. This is impossible to generate physically so that approximations must be used. The second is the step. F is then a constant. The third is a ramp where f is proportional to t and the last is a sine curve where  $f = a \sin(t)$  and a is a

constant. Of course, f is not limited to these functions, but one or more of the above will usually adequately describe the response of the system to an external disturbance.

Close approximations to the solutions of ordinary differential equations can readily be obtained using either analog or digital simulations. There are a number of algorithms available for digital use. Two of the most popular are those named after Euler and Kutta-Runge. The Euler algorithm is less accurate than the Kutta-Runge, but is simpler and requires less computation. The accuracy of the Euler method is normally sufficient for engineering use if the integration interval is small enough. The use of small integration intervals has one drawback and that is the mass of data generated. The obvious way to circumvent this problem is to plot the results. Fortunately, plotting is easy under COMAL.

I have written the program "dif'eq'plot.2" which will solve any ordinary differential equation, linear or non-linear up to and including the fourth order. The program can easily be extended to higher orders if required. The program uses Euler integration. I have checked the accuracy for first and second order equations with constant coefficients against the analytical solutions and found that errors do not exceed 4%. The solutions are plotted on the screen with automatic scaling. If, during any run, the x value goes off screen, the x scale is doubled and the equation resolved. The t scale is set when the end value of t is entered. The integration interval is set at t/1000.

More 

The entire program is straight forward except for the method of entering the coefficients. If we were going to limit the solutions to equations having constant coefficients, simple INPUT statements would suffice. However, when variable coefficients are used, data entry becomes more complex. The entry of variable coefficients during the program execution means changing the program. This necessitates a prepass before continuing execution. A RUN command performs a prepass, but it also performs a re-initialization of the variables which puts us back to square one with no coefficients. The problem can be circumvented by performing a SCAN in direct mode followed by a call to the appropriate proc. The SCAN is obtained by pressing the F8 key. The coefficient entry consists of adding two or more program lines complete with line numbers. Instructions are shown on the screen during the run. Following a run, pressing the F7 key will return to the menu.

Here are some equations you may want to try:

$$x' = -x + 1$$

Use 6 as the end value of t.

$$x'' = -x' - x + \sin(t)$$

Use 20 as the end value of t.

$$x''' = -2.25 x'' - 1.4 x' - 2 x + 1$$

Use 20 as the end value of t.

$$x'''' = -3.5 x''' - 5.5 x'' - 4.5 x' - 3x + \sin(t)$$

Use 20 as the end value of t.

In each case the last term is the forcing function, f.

The display will be functions f and x plotted against t. The f curve will be a dashed line and the x curve will be a continuous line.

Have fun!! ■

## PROTECTION PROGRAM UNBROKEN

If it can be protected, it can be broken. We know that. However, the COMAL 2.0 Cartridge PROTECT64 program protection system remains unbroken to this date. Here's a challenge for all the protection busters! Just get a COMAL 2.0 Cartridge and the Cartridge Demo Disk #3. The PROTECT64 program is on this disk. Of course it is protected. Can anyone send us a listing of it? A couple people have already tried, and declared that it is impossible. Should we believe them? And if they are right, then the commercial programmers should be very interested in COMAL 2.0 programming now!

## 2.0 FIND LAST PROGRAM LINE

Many times you may wish to know what the last line in your program is. Of course, you can LIST your program and wait while it scrolls past finally showing the last line. Or, just issue the command:

AUTO

Now hit the STOP key. The last line in the program is 10 less than the line prompted by the AUTO command.

## WARNING - CARTRIDGE USERS

You know that you are not supposed to plug in OR pull out the COMAL Cartridge while the computer is on. But, did you know that turning off the computer is not enough! You also must make sure the disk drive is turned off, for it shares the same lines with the cartridge. Don't blow it. Turn everything off when plugging the cartridge in. And once it's in, leave it in. To return the computer back the way it was before, just issue the command: BASIC.



# NOW COMAL 0.14 ON CARTRIDGE!

Now for only \$35.00 you can get  
COMAL 0.14 on cartridge! Turn on  
your C-64 and have comal in seconds!  
Don't waste time WAITING for Comal  
to load in! Get a 0.14 cartridge!

Send \$35 plus \$2 shipping to:  
Peripherals Plus  
4781 Windsong Ave.  
La Palma, CA 90623

CA residents add \$2.10 sales tax

Thirty day money back GUARANTEE!

Try it for 30 days and if for  
ANY reason you decide you do not  
want it just return the undamaged  
cartridge for a full refund on the  
purchase price!

## Join Us Today

NO GROUP OFFERS SO MUCH



### COMMODORE 64 & VIC-20 USERS

Join the largest VIC-20 / COMMODORE 64 users group in the United States.

#### MEMBERS RECEIVE:

- 10 Issues "Command Performance"
- Access to hundreds of VIC-20 and C64 public domain programs
- Technical assistance
- Informative reviews
- Contests
- Consumer assistance bureau
- Product purchasing assistance

Individual and chapter support  
12 month membership:  
U.S. \$20.00 - USA & Canada  
U.S. \$30.00 - Foreign

#1

**UNITED STATES COMMODORE USERS GROUP**  
**P.O. BOX 2310, Roseburg, OR 97470 USA**

**SPECIAL OFFER**  
**\$7.00**  
(NORMALLY \$19.95)

# Seeing Is Believing

---

"I don't have enough time or space to list all the good points!" -- *Noland Brown, MIDNITE SOFTWARE GAZETTE*

"This disk is fantastic!" -- *Tom Lynch, THE USERS PORT*

"Why all the enthusiasm? Because **COMAL** is a composite of the best features of the most popular programming languages... the familiarity of BASIC commands with the structural programming environment of Pascal and the turtle graphics of Logo." -- *Mark Brown, INFO 64*

"**COMAL** was just what I was looking for." -- *Collin Thompson, RUN*

---

Seeing **is** Believing. Take a look at what **COMAL** has to offer:  
the complete **COMAL 0.14 System** for Commodore 64™ includes the  
**Tutorial Disk\*** (*teaches you the fundamentals of COMAL*), plus  
the **Auto-Run DEMO Disk\*** (*demonstrates 26 COMAL programs  
including games, graphics, sprites and sounds*),  
all for just **\$7.00!**

You can add the reference book, *COMAL from A to Z*,  
for just **\$4.00 more.**

**\$7 or \$11** – either way you're a winner!

---

"Everybody who gets it, likes it! (I'll guarantee it.)" -- *Len Lindsay, President, COMAL Users Group*

---

Call **TOLL-FREE: 1-800-356-5324 ext. 1307**  
**VISA or MasterCard Orders ONLY.**  
**Questions and Information must call our**  
**Info Line: 608-222-4432.**  
**All orders prepaid only – no C.O.D.**



**Send check or money order in US Dollars to:**

**COMAL USERS GROUP, U.S.A., LIMITED**

5501 Groveland Ter., Madison, WI 53716  
phone: (608) 222-4432

\*Programs will come on 2 disks or 1 double sided disk -- each disk includes COMAL.  
Commodore 64 is a trademark of Commodore Electronics

## JOYSTICK CURSOR

by Mike Erskine

This program will move the cursor around the screen in the direction of the joystick, and leave a trail of periods (.) after it.

```
// delete "0:joy'cursor"
// save  "0:joy'cursor"
// by Mike Erskine
//
USE joysticks
USE system
textcolors(11,15,0)
joy'cursor(1,39,1,24)
//
PROC joy'cursor(left,right,top,bottom) C
LOSED //wrap line
//
USE joysticks
USE system
PAGE
LOOP
    fire:=FALSE
    row#:=currow
    column#:=curcol
    joystick(2,direction,fire)
    CASE direction OF
        WHEN 1
            row#:-1
        WHEN 2
            row#:-1
            column#:+1
        WHEN 3
            column#:+1
        WHEN 4
            row#:+1
            column#:+1
        WHEN 5
            row#:+1
        WHEN 6
            row#:+1
            column#:-1
        WHEN 7
            column#:-1
        WHEN 8
            row#:-1
            column#:-1
        OTHERWISE
            NULL
    ENDCASE
```

```
//
IF column#>right THEN
    column#:=right
ELIF column#<left THEN
    column#:=left
ENDIF
IF row#>bottom THEN
    row#:=bottom
ELIF row#<top THEN
    row#:=top
ENDIF
//
IF NOT fire THEN
    PRINT "144".",
    CURSOR (row#),(column#)
ELSE
    PAGE
ENDIF
PRINT "31""166""
CURSOR (row#),(column#)
ENDLOOP
ENDPROC joy'cursor
```

## COMAL 2.0 FUNCTION KEYS NOTE

The COMAL Cartridge allows you to define the function keys to whatever you want them to be. However, it appears that a function key cannot include a call to another function key. The function keys are not recursive.

## FUNCTION KEY DEFINITION LIMIT

The COMAL Cartridge allows you to define the function keys to be anything you'd like. However, it appears that a maximum of 32 characters is the limit on this definition.

## 2.0 MAXIMUM INPUT FIELD LENGTH

It appears that the maximum length your INPUT field can be is 120. This is done with the following:

```
INPUT AT 0,0,120: "Testing:":reply$
```

## RESTORE KEY DISABLE

This procedure will disable / enable the RESTORE key. This is useful during critical disk access.

```
PROC restore'key(state) CLOSED
  nmi'hi:=793; nmi'lo:=792
  IF state THEN
    POKE $03ff,$40 //rti instruction
    POKE nmi'hi,$03
    POKE nmi'lo,$ff
  ELSE
    POKE nmi'hi,$fe
    POKE nmi'lo,$47
  ENDIF
ENDPROC restore'key
```

## INDENTED PROGRAM TO DISK

If you LIST a program to disk in COMAL 0.14, the structures are not indented. If you would like to list the program to disk WITH indentations do this:

```
OPEN FILE 255,"TEST.L",UNIT 8,WRITE
SELECT OUTPUT "LP:"
LIST
```

You just tricked COMAL into thinking that the disk was a type of printer, so it listed to program with indentations.

## GEMINI SG-10C PRINTER

This printer is said to be Commodore 1525 compatible with no interface needed. It should work with the built in screen dumps from the COMAL 2.0 Cartridge.

## SPECIAL CHR\$ RESULTS

To switch to lower case mode:

```
PRINT CHR$(14),
```

To switch to upper case mode:

```
PRINT CHR$(142),
```

To lock the case (disable switching):

```
PRINT CHR$(8),
```

To unlock the case mode:

```
PRINT CHR$(9),
```

## PETSCII TO ASCII

You should be aware that Commodore uses a modified version of ASCII, sometimes referred to as PETSCII (named for their first PET computer). A reader wants us to remind you about the following three characters:

To get the ASCII underline use the LEFT ARROW key.

To get the ASCII backslash use the british pound sign key.

To get the ASCII caret use the UP ARROW key.

## TECH SKETCH LIGHT PEN

A reader has told us that the button on this light pen can be read by COMAL 2.0 as joystick direction 2.

**TODAY DISK 7****2.0 SIDE**

```

hi
>---programs---<
1520demo1
1520demo2
1520demo3
1520demo4
1520demo5
1520demo6
1520demo7
1520demo8
1520demo9
1520demo10
1520demo11
1520demo12
1520demo13
1520demo14
1520freeway
1520orbit'circle
1520sphere'plot
1520test'print
80'column'demo
ahl's-benchmark
batch'example
batchfile'maker
beams
calculart2
check'all'carts
cloud'flux
convert'listing
demo.fkeys
diff'equations
disassembler
disk'editor
draw'heart
find'radical
flow
freeway
gem10x'card/a
input'on'input
joy'cursor
k-s'stat'test
make'data'stmts
make'object'file
multiplication
perform'demo1
perform'demo2
polygons
read'directory
prnt'lrg'chars
sound.sample
test'external
>--procedures--<
func.get'protect
func.menu
func.modem'get$
proc.1520driver
proc.blankscreen
proc.comal-802
proc.load'font

```

```

proc.restore
proc.use'extend
>--data-files--<
bat.1520'40-list
bat.1520'80-list
bat.link-802dump
dat.information
font.80column
font.roundset
font.typeset
>---external---<
ext.double
ext.make'double

```

**TODAY DISK 7****0.14 SIDE**

```

boot c64 comal
c64 comal 0.14
hi
menu
>---programs---<
1520/0.14demo1
1520/0.14demo2
1520/0.14demo3
1520/0.14demo4
1520/0.14demo5
1520/0.14demo6
1520/0.14demo7
1520/0.14demo8
1520/0.14demo9
1520/0.14demo10
1520/0.14demo11
1520/0.14demo12
1520/0.14demo13
1520/0.14demo14
1520orbit'circle
1520sphere'plot
1520test'print
1541'alignment
1541'alignment1
1541'alignment2
add'errors
comal'program
eliza
freeway
gem10x'card/a
input'on'input
mailing'list
sort'all
txt'dump'ctl-p
>--procedures--<
1520/driver.proc
blankscreen.proc
get'protect.func
graphics.proc
menu.proc
show'error.proc
use'extend.proc
>--data-files--<
-error-messages-
information.dat

```

```

move'basic-$9000
sample.dat
sample.dat-1
sample.dat-2
sample.dat-3
sample.dat-4

```

**2.0 PACKAGES**

```

demo'1
demo'2
cfname'demo
link'binary'demo
print'symbols
show'libraries
>-symbol'tables<
c64symb
c64symb.merlin
c64symb.pal
syms
syms.mae
>--source'code-<
src.errorpack
src.example
>-package'files<
pkg.errorpack
pkg.example
>-binary'files-<
bin.example
>--ml'monitor--<
smon-comal
>--procedures--<
proc.link'binary
>--data'files--<
syms'alpha'order
syms'num'order

```

**MODEM DISK**

```

>--comal 0.14--<
>modem programs<
>-----<
terminal
vt-52.v4
>-----<
>-the following<
>-programs load<
>--from basic--<
>-----<
>-midwest term-<
>-----<
midwestterm5.1
term.c1
mlmid
>-----<
>-ravics term--<
>-----<
ravics term8.4
>--<
>--comal 2.0--<
>--for modem--<
>-----<
proc.modem
terminal'2.0

```

```

>- -----<
>-more basic----<
>- -----<
xmodem-auto
xmodem/autodial
xmodem/ml
secondterm
second.docs
2dtermboot
USER GROUP 8
bootcomal
c64 comal 0.14
em
comalerrors

```

**education**

```

demo'select'sort
selection'sort.l
demo'bubblesort
bubblesort.l
demo'insert'sort
insertion'sort.l
introd'quicksort
demo'quicksort
quicksort'vert.l
sort'timer'prg
rnd'name$'1000
demo'bin'search
binary'search
for'loop'part'1
for'loop'part'2

```

**utilities**

```

load'dump'epson
dump'epson
dump'mps801
uplots.l
three'd.l
geometry.l

```

**database**

```

filing
1541 database

```

**engineering**

```

section
reliability

```

**instructions**

```

read'about'disk
about'disk
file'to'print
letter

```

v1.1 06/02/85

Name: \_\_\_\_\_

(5/85)

Street: \_\_\_\_\_

City: \_\_\_\_\_ State: \_\_\_\_\_ ZIP: \_\_\_\_\_

NOTE: Pay by check, money order in US Dollars drawn on US B k. Canada Postal US Dollar Money Orders are OK. VISA/MasterCard: PRINT card #, exp date below with your signature:

VISA / MC number: \_\_\_\_\_ exp date: \_\_\_\_\_

Charge Orders: Signature: \_\_\_\_\_

>>> To qualify for subscriber prices we need your subscriber number: \_\_\_\_\_

>>> You automatically qualify if you start a new subscription now.

QNTY PRICE ITEM DESCRIPTION and LIST PRICE (all disks Commodore 1541 format)

**SYSTEMS:**

- [ ] \_\_\_\_\_ Deluxe Cartridge Package, \$128.90 plus \$3 shipping (2 books/2 disks/1 cartridge)
- [ ] \_\_\_\_\_ COMAL 2.0 Cartridge, \$99.95 plus \$2 shipping (no manual or disks)
- [ ] \_\_\_\_\_ C64 COMAL 0.14 Starter Kit, \$29.95 plus \$2 shipping
- [ ] \_\_\_\_\_ C64 COMAL 0.14 **\$11 SPECIAL** AutoRun & Tutorial disk, COMAL FROM A TO Z book
- [ ] \_\_\_\_\_ IBM PC COMAL from IBM Denmark (PC-DOS / MS-DOS) (Approx. \$250 - waiting for price)

**SUBSCRIPTIONS:**

- [ ] \_\_\_\_\_ **COMAL TODAY newsletter** >>> How many issues? \_\_\_\_\_ Start with # \_\_\_\_\_  
(\$14.95 for first 6 issues - \$2 each added issue) - (or \$2 for sample issue)
- [ ] \_\_\_\_\_ **TODAY DISK subscription** >>> How many? \_\_\_\_\_ Start with # \_\_\_\_\_  
(\$49.95 for first 6 disks - \$5 each added disk)

**BOOKS: (shipping \$2 each)(matching disks are \$19.95 or \$4.95 to subscribers)**

- [ ] \_\_\_\_\_ COMAL HANDBOOK \$18.95 - optional disk [ ]
- [ ] \_\_\_\_\_ FOUNDATIONS WITH COMAL \$19.95 - optional disk [ ]
- [ ] \_\_\_\_\_ STRUCTURED PROGRAMMING WITH COMAL \$26.95 - optional disk [ ]
- [ ] \_\_\_\_\_ BEGINNING COMAL \$22.95 - optional disk [ ]
- [ ] \_\_\_\_\_ COMMODORE 64 GRAPHICS WITH COMAL \$14.95 - disk not released yet
- [ ] \_\_\_\_\_ COMAL FROM A TO Z \$6.95 (no disk)
- [ ] \_\_\_\_\_ CAPTAIN COMAL GETS ORGANIZED (with disk) \$19.95 or \$12.95 to subscribers
- [ ] \_\_\_\_\_ COMAL WORKBOOK \$6.95 (no disk)
- [ ] \_\_\_\_\_ COMAL LIBRARY OF FUNCTIONS & PROCEDURES (with disk) \$19.95 or \$12.95 to subscribers
- [ ] \_\_\_\_\_ GRAPHICS PRIMER (with disk) \$19.95 or \$12.95 to subscribers
- [ ] \_\_\_\_\_ CARTRIDGE GRAPHICS AND SOUND \$9.95 or \$6.95 to subscribers
- [ ] \_\_\_\_\_ COMAL 2.0 PACKAGES (disk includes C64SYMB & MONITOR) \$19.95 or \$12.95 to subscribers
- [ ] \_\_\_\_\_ CARTRIDGE TUTORIAL BINDER \$15.00

**DISKS: (NOTE: all disks below are only \$9.75 each to subscribers!).**

- [ ] \_\_\_\_\_ 19 DISK SET, \$94.05 (about 1000 programs for COMAL 0.14)
- [ ] \_\_\_\_\_ BEST OF COMAL, \$19.95 (\$9.75 to subscribers)(includes Disk Data Base)
- [ ] \_\_\_\_\_ C64 COMAL SAMPLER, \$19.95 (\$9.75 to subscribers)(part of Starter Kit)
- [ ] \_\_\_\_\_ AUTO RUN DEMO DISK and TUTORIAL DISK, **\$7.00** (part of Starter Kit)
- [ ] \_\_\_\_\_ Bricks TUTORIALS (2 sided BEGINNERS disk!), \$19.95 (\$9.75 to subscribers)
- [ ] \_\_\_\_\_ Modem Disk \$14.95 (9.75 to subscribers)(for COMAL 0.14 & 2.0).
- [ ] \_\_\_\_\_ UTILITY DISK, \$19.95 (\$9.75 to subscribers)
- [ ] \_\_\_\_\_ TODAY DISK, \$14.95 (\$9.75)>>>> **Which one(s):** \_\_\_\_\_
- [ ] \_\_\_\_\_ USER GROUP DISK, \$10 (\$9.75)>>>> **Which one(s):** \_\_\_\_\_
- [ ] \_\_\_\_\_ CARTRIDGE DEMO DISKS, \$19.95 (\$9.75)>>>> **Which one(s):** \_\_\_\_\_

**OTHER**

- [ ] \_\_\_\_\_ OTHER: \_\_\_\_\_
- [ ] \_\_\_\_\_ KEYBOARD OVERLAY for C64 COMAL 0.14 - CHEATSHEET, \$3.95 (plus \$1 handling)
- [ ] \_\_\_\_\_ McPen Light Pen, \$49.95 (with COMAL 2.0 Demo Disk)



TOTAL \_\_\_\_\_ + \_\_\_\_\_ Shipping = TOTAL PAID \$ \_\_\_\_\_ (WI add 5% sales tax)

Mail To: COMAL Users Group USA, 6041 Monona Drive, Madison, WI 53716 or call 608-222-4432

Yes, I want COMAL TODAY. Give me \_\_\_\_\_ issues.  
(\$14.95 for 1st 6, \$2 each after that)

This is a:        ☐ New Subscription

☐ Renewal

My subscriber # is: \_\_\_\_\_

Name: \_\_\_\_\_

Address: \_\_\_\_\_

City: \_\_\_\_\_ State: \_\_\_\_\_ ZIP: \_\_\_\_\_

☐ Check enclosed        ☐ Visa        ☐ MasterCard

Card # \_\_\_\_\_ expires \_\_\_\_\_

Signature: \_\_\_\_\_



**From:**

**PLACE  
STAMP  
HERE**

**TO: COMAL USERS GROUP, USA, LIMITED  
5501 GROVELAND TERRACE  
MADISON, WI 53716-3251**

---

# Subscriber's Specials

---

Here are this issues specials for subscribers only. If you subscribe now, you can take advantage of these specials at the same time. If you are a current subscriber, you **MUST** include your subscriber number with your order, or we will not honor the special prices. Your subscriber number is printed on the newsletter mailing label. New subscribers get these prices automatically and a number will be assigned with the order.

## \$25 FREE COMAL 2.0 ITEMS

When you order the Cartridge or Deluxe Cartridge Pack, our order processing system will apply a \$25 credit to your order. Order any extra books or disks you want and \$25 will be deducted from their cost!

## CAPTAIN COMAL BOOK SPECIAL

CAPTAIN COMAL GETS ORGANIZED, LIBRARY OF PROCEDURES AND FUNCTIONS, GRAPHICS PRIMER, and COMAL 2.0 PACKAGES book/ disk sets, usually \$19.95, now only \$12.95 each set. Plus, CARTRIDGE GRAPHICS AND SOUND, usually \$9.95, now only \$6.95.

## C64 GRAPHICS WITH COMAL BOOK

The graphics reference book for COMAL 0.14, usually \$17.95, now only \$14.95.

## ANY COMAL DISK ONLY \$9.75

Now you can get any COMAL disk for only \$9.75 - including our double sided disks. Or get our 19 disk boxed set for only \$94.05. Of course this does not include those disk mentioned below at an even lower special price.

## ANY MATCHING DISK ONLY \$4.95

When you get one of the COMAL tutorial books, you can get it's matching disk at the same time for only \$4.95.

## DISK SUBSCRIPTION ONLY \$49.95

Subscribe to the TODAY disks now at a special price. These disks are double sided begining with TODAY DISK #6. Six disks (12 sides!) for only \$49.95, and that includes the shipping! **EVEN MORE:** Extend your disk subscription beyond the usual 6 disks. Add onto your disk subscription as many disk issues as you wish for only \$5 each. That is only about \$2.50 per disk side! How can you refuse. Specify which disk to start with.

## NEWSLETTER SUBSCRIPTION

Renew your subscription now, \$14.95 for 6 issues. **EVEN MORE:** extend your renewal beyond the usual 6 issues. Add on as many more issues as you want for only \$2 each. Thus a 10 issue newsletter subscription would cost \$22.95 (\$14.95 for 6 issues plus \$8 for 4 added issues). Why worry about renewals in the future - get a 48 issue subscription now for only \$98.95 (prepare for when COMAL TODAY goes monthly - at a LOW price now). New subscribers make sure to specify the issue to start with.

## FREE PLAYNET SETUP OFFER

COMAL ONLINE SUPPORT is now available via PlayNET (see COMAL TODAY #6, page 46, for story). It normally costs \$39.95 to setup your PlayNET account (only \$2 per hour after that). Until July 31, 1985 we can set up your PlayNET account for FREE (see ad on page 17). We will send you the manual and complete PlayNET system disk FREE, you just pay the \$2 shipping cost. To qualify, just do one of two things:

\* Subscribe or renew a TODAY DISK subscription.

\* Subscribe or renew a COMAL TODAY newsletter subscription for **10 or more** issues.

**100,000** CHOOSE COMAL  
**50,000** USERS†

**(1) DISK BASED COMAL Version 0.14**

- COMAL STARTER KIT—Commodore 64™ System Disk, Tutorial Disk (interactive book), Auto Run Demo Disk, Reference Card and COMAL FROM A TO Z book. \$29.95 plus \$2 handling

**(2) PROFESSIONAL COMAL Version 2.0**

- Full 64K Commodore 64 Cartridge  
Twice as Powerful, Twice as Fast  
\$99.95 plus \$2 handling (no manual or disks)
- Deluxe Cartridge Package includes:  
COMAL HANDBOOK 2nd Edition, Graphics and Sound Book, 2 Demo Disks and the cartridge (sells for over \$200 in Europe). This is what everyone is talking about. \$128.90 plus \$3 handling (USA & Canada only)

**CAPTAIN COMAL™ Recommends:**

The COMAL STARTER KIT is ideal for a home programmer. It has sprite and graphics control (LOGO compatible). A real bargain—\$29.95 for 3 full disks and a user manual.

Serious programmers want the Deluxe Cartridge Package. For \$128.90 they get the best language on any 8 bit computer (the support materials are essential due to the immense power of Professional COMAL).

**ORDER NOW:**

Call TOLL-FREE: 1-800-356-5324 ext 1307 VISA or MasterCard ORDERS ONLY. Questions and information must call our Info Line: 608-222-4432. All orders prepaid only—no C.O.D. Send check or money order in US Dollars to:

**COMAL USERS GROUP, U.S.A., LIMITED**  
5501 Groveland Ter., Madison, WI 53716

TRADEMARKS: Commodore 64 of Commodore Electronics Ltd. Captain COMAL of COMAL Users Group, U.S.A., Ltd.  
† estimated

=====

**IMPORTANT NOTICE**

=====

**YOUR SUBSCRIBER NUMBER IS  
PRINTED ON YOUR MAILING LABEL**

**PLEASE INCLUDE THIS NUMBER  
WITH ANY FUTURE ORDERS OR  
CORRESPONDENCE**

**THANK YOU**

=====

**IMPORTANT NOTICE**

=====

**COMAL TODAY**  
**COMAL USERS GROUP, U.S.A., LIMITED**  
6041 Monona Drive, Madison, WI 53716

**BULK RATE  
U.S. POSTAGE  
PAID  
MADISON, WI  
PERMIT 2981**

**IF YOUR LABEL SAYS  
LAST ISSUE: 7  
YOU MUST RENEW NOW.  
SUBSCRIPTION CARD  
IS INSIDE.**